QTL Cartographer
Version 1.14

Christopher J. Basten Bruce S. Weir Zhao-Bang Zeng

June 22, 2000

QTL Cartographer

QTL Cartographer

A Reference Manual and Tutorial for QTL Mapping

Christopher J. Basten, Bruce S. Weir and Zhao-Bang Zeng
Program in Statistical Genetics
Department of Statistics
North Carolina State University

QTL Cartographer

Copyright ©2000 by Christopher J. Basten, Bruce S. Weir and Zhao-Bang Zeng.
Program in Statistical Genetics
Department of Statistics
North Carolina State University
Raleigh, NC 27695-8203.

All rights reserved. Reproductions for personal use are allowed. Anyone wishing to reproduce this book in whole or in part by any means for profit must first obtain permission from the authors.

Printed in the United States of America
Typeset in LaTeX2e on a Macintosh G3 using Textures version 1.8 from Blue Sky Research, Inc.

Contents

List of Figures **9**

List of Tables **11**

1 Introduction **13**
1.1 General Overview . . . 13
1.1.1 Definition of the Problem . . . 13
1.1.2 Experimental Design . . . 14
1.1.3 Genetic Linkage Maps . . . 16
1.2 Programming Philosophy . . . 16
1.3 Copyright Information and Acknowledgments . . . 16
1.3.1 QTL Cartographer Copyright Information . . . 16
1.3.2 Citing QTL Cartographer . . . 17
1.3.3 Gnuplot Copyright Information . . . 17
1.3.4 LINPACK Copyright Information . . . 18
1.3.5 Numerical Recipes in C Information . . . 18
1.4 How to Get and Install QTL Cartographer . . . 19
1.4.1 MS-Windows . . . 20
1.4.2 UNIX . . . 20
1.4.3 Macintosh . . . 21
1.5 Getting Help . . . 21
1.5.1 Mailing List . . . 22
1.5.2 Bug Reports . . . 22
1.5.3 Contacts . . . 23
1.6 General Usage of the Programs . . . 23
1.6.1 Options for all programs . . . 24
1.6.2 Filenaming Conventions . . . 28

2 Simulating/Reformatting Data **31**
2.1 Rmap . . . 32
2.1.1 Simulating a Map . . . 33
2.1.2 Using MAPMAKER/EXP files . . . 34
2.1.3 QTL Cartographer user input format . . . 34

2.1.4 Command Line Options . . . 35
2.2 Rqtl . . . 37
2.3 Rcross . . . 38
2.3.1 Simulating Data . . . 38
2.3.2 Translating Data . . . 40
2.3.3 Output . . . 40
2.4 Prune . . . 42
2.4.1 Pruning Datasets . . . 43
2.4.2 Recreating Datasets . . . 44

3 Analysis **49**
3.1 Qstats . . . 49
3.1.1 Command Line Options . . . 51
3.1.2 Segregation . . . 52
3.2 LRmapqtl . . . 53
3.2.1 Simple Linear Regression . . . 53
3.2.2 Output . . . 54
3.2.3 Permutation Tests . . . 54
3.3 SRmapqtl . . . 55
3.3.1 Output . . . 55
3.4 Zmapqtl . . . 56
3.4.1 Computational Methodology . . . 57
3.4.2 Models . . . 58
3.4.3 Zmapqtl Options . . . 59
3.4.4 Output . . . 61
3.5 JZmapqtl . . . 65
3.5.1 JZmapqtl Options . . . 65
3.5.2 Output . . . 66
3.5.3 Usage Hints . . . 66

4 Visualization of Results **69**
4.1 Eqtl . . . 69
4.1.1 Options . . . 70
4.2 Preplot . . . 72
4.2.1 Printing Results . . . 72
4.3 GNUPLOT . . . 73
4.3.1 Basic GNUPLOT . . . 74

5 Tutorial Examples **75**
5.1 General tactics and notes . . . 75
5.2 Basic Macintosh . . . 76
5.3 Basic Windows . . . 76
5.3.1 Navigating disks . . . 76
5.4 Basic Unix . . . 77

5.4.1 Help! . . . 77
5.4.2 Basic filesystem commands . . . 77
5.4.3 Curious? . . . 78
5.4.4 Other commands . . . 78
5.5 Simulating and Analyzing data . . . 79
5.6 Analyzing simulated data . . . 80
5.7 Analyzing real data . . . 80
5.8 Analyzing a MAPMAKER data set . . . 81
5.8.1 Using MAPMAKER/EXP . . . 81
5.8.2 Using the MAPMAKER files . . . 82

6 Input File Formats **85**
6.1 Genetic Linkage Maps . . . 85
6.1.1 MAPMAKER output files . . . 85
6.1.2 Rmap input files . . . 85
6.1.3 Rmap output files . . . 87
6.2 QTL information . . . 87
6.2.1 Rqtl input files . . . 87
6.2.2 Rqtl output files . . . 88
6.3 Data files . . . 89
6.3.1 MAPMAKER raw files . . . 89
6.3.2 Rcross input files . . . 89

7 Benchmarks **93**

8 UNIX Man Pages **95**
8.1 QTLCART . . . 96
8.2 RMAP . . . 102
8.3 RQTL . . . 106
8.4 RCROSS . . . 109
8.5 PRUNE . . . 112
8.6 QSTATS . . . 115
8.7 LRMAPQTL . . . 117
8.8 SRMAPQTL . . . 119
8.9 ZMAPQTL . . . 121
8.10 JZMAPQTL . . . 125
8.11 PREPLOT . . . 129
8.12 EQTL . . . 131

Bibliography **134**

Index **136**

List of Figures

1.1 Basic Cross 15

2.1 Reformatting Data 32
2.2 Simulating Data 32

3.1 Analysis Schematic 50

4.1 Visualization Schematic 70

List of Tables

1.1 Summary of Experimental Design Codes 15
1.2 Subroutines from *Numerical Recipes in C* 18
1.3 Contact for Help 23
1.4 Command Line Options for all programs 25
1.5 Standard Filename Extensions and File types for Output Files 29
1.6 Miscellaneous Files and File types 29

2.1 Command Line Options for Rmap 35
2.2 Command Line Options for Rmap 36
2.3 Command Line Options for Rqtl 37
2.4 Command Line Options for Rcross 38
2.5 Command Line Options for Prune 43

3.1 Command Line Options for Qstats 52
3.2 Command Line Options for LRmapqtl 54
3.3 Command Line Options for SRmapqtl 55
3.4 Command Line Options for Zmapqtl 59
3.5 Examples of Interim Files for Model 6 61
3.6 Command Line Options for JZmapqtl 65

4.1 Command Line Options for Eqtl 71
4.2 Command Line Options for Preplot 73
4.3 Filename extensions for Preplot output 73

7.1 Timings for Interval Mapping 93
7.2 Timings for Composite Interval Mapping 94

Chapter 1

Introduction

1.1 General Overview

QTL Cartographer is a suite of programs for mapping quantitative trait loci (QTLs) onto a genetic linkage map. The general experimental paradigm begins with a pair of inbred parental lines that differ in the trait of interest and in the set of marker genotypes. The programs use linear regression, interval mapping (Lander and Botstein 1989) or composite interval mapping (Zeng 1993; Zeng 1994) methods to dissect the underlying genetics of the quantitative traits. Mapping is done onto a set of linked genetic markers with known recombination frequencies. Genetic linkage maps and data files can be imported from **MAPMAKER** (Lander et al. 1987). The mapping program uses a dynamic algorithm that allows a host of statistical models to be fitted and compared, including various gene actions (additive and dominance), QTL-environment interactions, and close linkage.
This package consists of several programs written in C to perform various tasks, including simulating, reformatting or analyzing data and visualizing the results of the analyses. Presently, the mapping programs can handle data from backcrosses, intercrosses and recombinant inbreds, as well as a few other experimental designs (see Table 1.1).
All input and output files are plain text and can be viewed or imported into many text editors and graphics packages on various computing platforms. The programs were originally written for the UNIX operating system and have since been ported to the Macintosh and Microsoft Windows operating systems. Present development is on a Macintosh using Metroworks Codewarrior. Both Macintosh and Windows binaries are created using Metroworks. The UNIX distribution is of the source code. This project is ongoing and suggestions are welcome for further improvements and enhancements. The source code and compiled binaries are freely available and may be obtained by anyone over the internet.

1.1.1 Definition of the Problem

Often traits in plants and animals are influenced by many genes rather than a single locus (Falconer and MacKay 1996, for an excellent general review). These traits are termed quantitative traits and the loci that control these traits quantitative trait loci, abbreviated

henceforth as QTLs. An important goal in genetics and breeding is to identify and characterize QTLs, especially those that contribute to variation in quantitative traits both within and between populations or species. The recent advances in molecular biology have allowed the construction of genetic linkage maps based on molecular markers. Such genetic linkage maps can span the genome at regular intervals. The experimenter can then look for correlations between these mapped markers and the trait of interest in controlled breeding experiments to gain insight into the regions of the genome that control the trait.

1.1.2 Experimental Design

The paradigm for the programs in the ***QTL Cartographer*** package is that of highly inbred lines with very little genetic variation within lines but variation between lines. We shall refer to these inbred lines as parental lines and denote them by the symbols P_1 and P_2. As a general rule, the P_1 lines will correspond to the "high" lines with respect to the trait of interest, that is they will have mean values larger than the P_2 or "low" lines. These parental lines can be crossed to produce F_1 lines which are heterozygous for both markers and QTLs. One can then cross the F_1 populations with either parental line to produce backcrosses. The symbols B_1 and B_2 will refer to backcrosses involving the P_1 and P_2 lines, respectively. Alternatively, the F_1 lines can be intercrossed to produce F_2 lines.
In each of these cases, the resultant lines will have variation in both the trait of interest and the underlying quantitative trait loci and marker genotypes. These crosses are illustrated in Figure 1.1. We can then look for correlations between the trait in question and marker genes that have been mapped previously.
We have also included options for more complex experimental designs, including recombinant inbred lines, general F_t lines produced by selfing or random crossing of F_{t-1} lines, *etc.* The programs in the ***QTL Cartographer*** system will need to know the type of experimental design used to create the data. This design is encoded by a string of characters. If the letter i stands for some integer, then the possible crosses will be B_i, SF_i, RF_i, RI_i, $T(XX)SF_i$ and $T(XX)RF_i$. The *B* stands for a backcross and the integer attached to it will indicate the parental line to which the F_1 line was crossed to (either 1 or 2). If there was repeated backcrossing to one of the parental lines, this can be indicated by attaching two integers to the *B*: B_{ij} indicates that there were j generations of backcrossing to parental line i. B_{11} is equivalent to B_1. SF_i stands for selfed intercross lines and the integer indicates the generation ($i = 2,\ 3,\ \ldots$). RF_i stands for randomly mated intercross lines. *RI* means recombinant inbred lines, and the integer can take on one of three values: 0, 1, and 2. A 1 indicates *RI* lines derived by selfing, a 2 by sib mating and a 0 means doubled haploid lines.
The *T* indicates that the data are the result of a test cross. For a test cross, genotyping is done on an intercross (SF_i or RF_i) and phenotyping on a cross derived from that intercross. The first part of the string, *T(XX)* indicates that phenotyping is done on the *XX* population and the second part (SF_i or RF_i) indicates the genotyped population. *XX* can be a B_1, B_2, SF_i or D_3 for SF_i lines or B_1 or B_2 for RF_i lines. D_3 stands for Design III experiments (Cockerham and Zeng 1996).
All of the above experimental designs can be simulated, and all but the Design III ex-

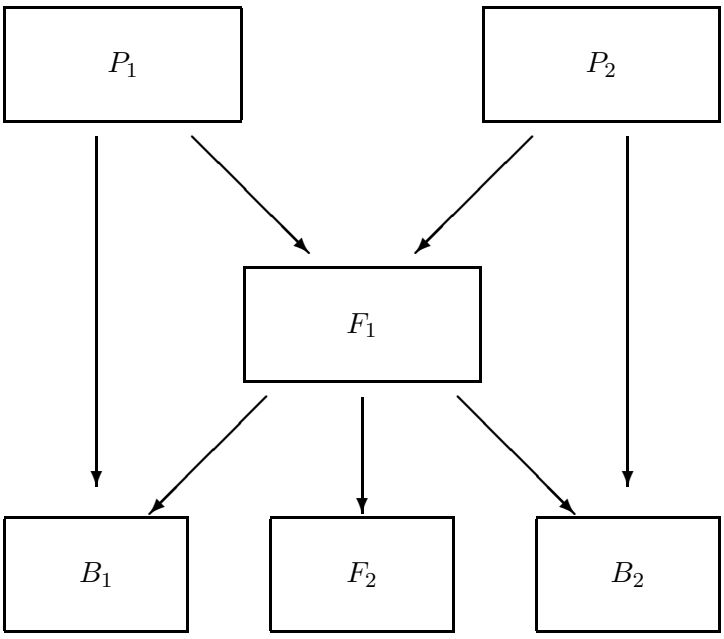

Figure 1.1: Basic Cross

Design	Code	Example
Backcross to P_i	B_i	B1
Backcross j times to P_i	B_{ij}	B13
Selfed generation i intercross	SF_i	SF3
Randomly mated generation i intercross	RF_i	RF2
Doubled Haploid	RI_0	RI0
Recombinant Inbred via selfing	RI_1	RI1
Recombinant Inbred via sib mating	RI_2	RI2
Testcross of SF_i to P_j	$T(B_j)SF_i$	T(B1)SF3
Testcross of SF_i for j generations	$T(SF_{i+j})SF_i$	T(SF4)SF3
Testcross of RF_i to P_j	$T(B_j)RF_i$	T(B1)RF3
Design III	$T(D3)SF_i$	T(D3)SF5

Table 1.1: Summary of Experimental Design Codes

periments can be analyzed. Table 1.1 lists all the experimental designs and their ***QTL Cartographer*** codes. The experimental designs of Table 1.1 can be specified in Rcross for simulations or in certain data input files (see Section 6.3.2).

1.1.3 Genetic Linkage Maps

A known genetic linkage map will be required for the analysis. A good genetic linkage map will comprise a set of Mendelian marker loci that are evenly spaced and span the genome. Average intermarker distances of 5 to 10 centimorgans would be optimal. We have provided ways to simulate linkage maps as well as to convert linkage map information into a format suitable for ***QTL Cartographer*** . Presently the user has two options for genetic linkage map input. The first is a format designed for the ***QTL Cartographer*** system that allows for free annotation of the data file. An example is given in (6.1.2). A second option allows the user to import the results of a **MAPMAKER** session. This is covered in more detail in (2.1) and (5.8.1).

1.2 Programming Philosophy

These programs were originally developed on a UNIX workstation. Consequently, the programming philosophy is heavily influenced by the UNIX operating system. All the programs have command line options which mimic those of regular UNIX commands. We have added interactive menus so as to make the programs more user friendly on Macintoshes and PCs running Microsoft Windows^{TM}.

There are a number of different programs in the package rather than one program that does everything. In this way, each program does a small job, and the user can combine the programs as a group to do a complete analysis. The user can examine the input and output files for each step and have a better idea of what the programs are doing. All input and output files are plain ASCII text. They can be transferred to any platform and viewed or edited there.

We have also been influenced by the Free Software Foundation in that we charge no fee for this program package. We have attempted to integrate these programs with other free software (most notably **GNUPLOT** and **MAPMAKER**).

1.3 Copyright Information and Acknowledgments

1.3.1 QTL Cartographer Copyright Information

Copyright (C) 1994-2000 C. J. Basten, B. S. Weir and Z.-B. Zeng

Permission to use, copy, and distribute this software and its documentation for any purpose with or without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

Permission to modify the software is granted, but not the right to distribute the modified code. Modifications are to be distributed as patches to released version.

This software is provided “as is” without express or implied warranty.

1.3.2 Citing QTL Cartographer

In publications, you should cite our original short announcement (Basten, Weir, and Zeng 1994) and this manual.

- C. J. Basten, B. S. Weir and Z.-B. Zeng, 1994. Zmap–a QTL cartographer. IN *Proceedings of the 5th World Congress on Genetics Applied to Livestock Production: Computing Strategies and Software*, edited by C. Smith, J. S. Gavora, B. Benkel, J. Chesnais, W. Fairfull, J. P. Gibson, B. W. Kennedy and E. B. Burnside. Volume 22, pages 65-66. Published by the Organizing Committee, 5th World Congress on Genetics Applied to Livestock Production, Guelph, Ontario, Canada.

- Basten, C.J., B.S. Weir and Z.-B. Zeng, 2000. QTL Cartographer, Version 1.14. Department of Statistics, North Carolina State University, Raleigh, NC.

1.3.3 Gnuplot Copyright Information

We suggest that you download and make use of the fine plotting package **GNUPLOT** (Williams and Kelley 1993). which we use as the graphics engine to display the results of analyses. **GNUPLOT** is freely available for UNIX, Macintosh and MS-Windows machines. It is quite easy to use, produces nice results and all the input files are plain text. We reprint the copyright information for **GNUPLOT** verbatim:

```
GNUPLOT copyright information:

  Copyright (C) 1986 - 1993   Thomas Williams, Colin Kelley

Permission to use, copy, and distribute this software and its
documentation for any purpose with or without fee is hereby granted,
provided that the above copyright notice appear in all copies and
that both that copyright notice and this permission notice appear
in supporting documentation.

Permission to modify the software is granted, but not the right to
distribute the modified code.  Modifications are to be distributed
as patches to released version.

This software is provided "as is" without express or implied warranty.

AUTHORS

 Original Software:
   Thomas Williams,  Colin Kelley.

 Gnuplot 2.0 additions:
     Russell Lang, Dave Kotz, John Campbell.

 Gnuplot 3.0 additions:
```

```
     Gershon Elber and many others.

For more information on GNUPLOT, see the documentation
that comes with the program.
```

1.3.4 LINPACK Copyright Information

We have translated some of the FORTRAN procedures of LINPACK (Dongarra et al. 1979) into C. We have used all of the basic linear algebra subroutines (BLAS) as well as subroutines to do the QR factorization of the matrix $\mathbf{X}$ of the linear system

$$\vec{y} = \mathbf{X} \cdot \vec{b}$$

These include the subroutines SQRST, STRSL, SPODI, SQRSL and SQRDC. Not all of the optimizations have been translated. These subroutines are used quite extensively in the analysis modules. The original FORTRAN subroutines are Copyright (C) 1979 by the Society for Industrial and Applied Mathematics.

1.3.5 Numerical Recipes in C Information

We have made extensive use of the ideas from *Numerical Recipes in C* (Press, Flannery, Teukolsky, and Vetterling 1988). The source file "Utilities.c" contains subroutines for allocating memory that are derived from the functions for creating arbitrary offset vectors and matrices in Appendix D. We have also used modified versions of the subroutines listed in Table 1.2. The original subroutines are Copyright (C) 1987, 1988 *Numerical Recipes Software.*

subroutine	section
indexx()	8.3
moment()	13.1
sort()	8.2
gammln()	6.1
gammp()	6.2
gasdev()	7.2
gcf()	6.2
gser()	6.2
poidev()	7.3
betai()	6.3
beta()	6.1
betacf()	6.3

Table 1.2: Subroutines from *Numerical Recipes in C*

1.4 How to Get and Install QTL Cartographer

Point your web browser to "http://statgen.ncsu.edu" and follow the link to software, and from there to ***QTL Cartographer*** . You can then follow the link to the ftp site and shift-click on the files you want to download.
QTL Cartographer is also downloadable via anonymous ftp at *statgen.ncsu.edu* (152.1.95.36). Use "ftp" as your username and your email address as the password. Here is an example.

```
username: ftp
password: basten@statgen.ncsu.edu
```

Next, change directory into the distribution subdirectory, **/pub/qtlcart**, and view what is available. For example,

```
ftp> cd /pub/qtlcart
ftp> ls
ChangeLog
QTLCartMac.sea.hqx
QTLCartWin.zip
QTLCart.tar.Z
README
gnuplot.exe
gnuplot.sit.hqx
gnuplot.tar.Z
1.10b/
1.12f/
1.13g/
swang/
```

Download the appropriate version. Presently, the following versions are available.

- **QTLCartWin.zip** is for Microsoft Windows. These are 32 bit applications. You will need an unzip utility to unpack this file. There should be an unzip utility in the Windows system folder.
- **QTLCartMac.sea.hqx** is for Macintoshes
- **QTLCart.tar.Z** is for UNIX
- *gnuplot* files are the distributions of **GNUPLOT** for various platforms
- The *1.10b, 1.12f* and *1.13g* are directories containing older versions of ***QTL Cartographer***
- The *swang* directory contains Shengchu Wang's Windows front-end for ***QTL Cartographer***

You can usually download a file by using the **get** command with a filename. On Macintoshes, using the server mode may require you to use the **put** command, as you are putting the files onto your local machine rather than getting them from the remote server. It is best to do the transfer in an empty subdirectory so that you don't inadvertently delete some important files. You will also want to download the *README* file if you don't already have a copy of it. The *README* file in the **/pub/qtlcart** subdirectory will often be more recent than the one in the archive.
The **manual.pdf** and **manpages.pdf** files are Adobe Portable Document Files of the manual and the UNIX manpages. The manual is the present document, and the manpages are meant to be appended to this document. You can view or print these files with Adobe Acrobat Reader, which is freely available from the Adobe website (http://www.adobe.com).
The following sections indicate how to install the programs onto various computing platforms.

1.4.1 MS-Windows

Download the file **QTLCartWin.zip** in binary format to your computer's hard drive. Move the program to a directory where you want ***QTL Cartographer*** to reside. Use the program unzip utility to unpack the MS-Windows distribution (this can be done with a double-click).
The programs will be unpacked in the directory you choose. You will want to do this in a directory created for ***QTL Cartographer*** , so let's assume that it is **c:\qtlcart**.
You may also want to download **GNUPLOT** for MS-Windows. In binary format, get the self-extracting archive **gnuplot.exe**. Put it in a subdirectory (say **C:\gnuplot**) and while in that subdirectory, run **gnuplot** from the DOS command line.
The programs can be run by double clicking their icons in the **Windows Explorer** application. An alternate method is to open a **Command Window** and type in the program names.
You can view the output files in any text editor, although you should be aware that some editors in MS-Windows cannot load large files.

1.4.2 UNIX

Download the file **QTLCart.tar.Z** in binary form from *statgen.ncsu.edu*. It is in the same directory that **README** file came from. On your local machine, create a subdirectory for the distribution, then move the file **QTLCart.tar.Z** to it. Uncompress and untar the file as follows:

```
% uncompress QTLCart.tar.Z
% tar xf QTLCart.tar
```

Follow these steps to compile and install ***QTL Cartographer*** .

1. Move into the **src** directory and copy the file **LocalD.h.UNIX** to **LocalD.h**. This file may be fine for your system. It is annotated and you can follow the directions in the file if compilation doesn't work the first time.

2. You will also need to edit the **Makefile** and choose a compiler. The default is **gcc**, which is the compiler used on our Sun workstations running Solaris. If you don't have **gcc**, you might try **cc**. Finally, you will want to set the install directory. By default it is BINDIR = /usr/local/bin, but you can change it to whatever you wish. Note that to install the programs in the install subdirectory, you will need write permissions for that subdirectory.

3. Change into the root directory of the distribution and make the programs:

   ```
   % make install
   ```

4. The binaries will be in the BINDIR subdirectory. Make sure that this subdirectory is in your path variable, and then **rehash**.

Presently, we use **gcc** version 2.95.2 on our Sun Workstations under Solaris 2.7. If you have troubles compiling, you may need to update your operating system or compiler.
If you would like a hardcopy of the man pages, you can either **cd** to the **doc** subdirectory and send the postscript files to a postscript printer, or **cd** to the **man** subdirectory and issue the **make hardcopy** command. The second method requires that the program **a2ps** be installed on your system, and that the default printer be able to handle postscript. Alternatively, you can print the files ending in "pdf" using **Adobe Acrobat Reader,**, which is freely available from Adobe (http://www.adobe.com).

1.4.3 Macintosh

You will need a Macintosh with a power pc chip. Download the file **QTLCartMac.sea.hqx**. Use **StuffitExpander** or **BinHex4** to unbinhex the self-extracting archive. Double click the **QTLCart.sea** file to unpack the binaries and supplemental files. Some programs such as **Netscape** or **Fetch** will unbinhex the files for you, although they may require a helper application.
Once the file **QTLCart.sea.hqx** has been unpacked, you will have a folder called **bin** with the programs in it. You can simply double click on any of them to start them up. You will first be presented with a console window. All you need to do is click on **OK** to get to the interactive menu for setting options. Note that when you double click on a ***QTL Cartographer*** program, you will get a command line interface window: You can simply click "OK" here to get the menu. You can also enter command line options in that box if you like.

1.5 Getting Help

One of the best places to get help is from the UNIX man pages. These should be installed with the UNIX distribution of the program, and are described in the APPENDIX. (Since Macintosh and MS-Windows users won't have the man pages, we have attached them to this document as an APPENDIX and included them in the **doc** subdirectory.) This document, as well as the man pages, are available via the World Wide Web by pointing your web browser to

`http://statgen.ncsu.edu/`

and following the link to ***QTL Cartographer*** , which is halfway down the page.

1.5.1 Mailing List

The address for the mailing list server is *MajorDomo@statgen.ncsu.edu*. Please join the mailing list for ***QTL Cartographer*** . It will be a forum for problems you may have in using the programs, and we will post announcements of updates and bug fixes. To subscribe, send the following two line message to the server:

```
subscribe qtlcart
end
```

The second line in the message stops MajorDomo from interpreting your .sig. Note that the subject line of your mail message will be ignored. If the subscription was successful, you will receive a confirmation note saying as much. You may also put an email address after the "subscribe qtlcart" (on the same line) to subscribe that address:

```
subscribe qtlcart basten@statgen.ncsu.edu
end
```

A message like the above with "unsubscribe" rather than "subscribe" would unsubscribe the address. The command "help" would cause the server to return a list of commands that can be sent to the MajorDomo server. Remember that all commands should be directed to *MajorDomo@statgen.ncsu.edu*, while messages for people on the list go to *qtlcart@statgen.ncsu.edu*.

1.5.2 Bug Reports

Send any bug reports to *qtlcart-bug@statgen.ncsu.edu*. There is certain information that will greatly aid in diagnosing the problem. The ***QTL Cartographer*** distribution should come with a file called **problems.txt** with the following questions in it:

1. Computing platform
   (a) What machine are you using? Is it a
      i. UNIX based workstation?
      ii. PC running Windows?
      iii. PowerPC based Macintosh?
   (b) What operating system is it running?
   (c) What is the version of the Operating system?
   (d) How much memory and free hard disk space do you have?
2. Programs

(a) Which program is giving you trouble, and what parameter values were used?

(b) Are the input files simulated or real?

(c) Would it be possible to send me the input files, the log file and the resource file (qtlcart.rc)?

(d) When the program crashed, did it give any diagnostics?

(e) When did you download the programs?

(f) What is the version number? (This is valid for programs downloaded after 1 January 1996, and supersedes the previous question.)

When reporting a problem, try to include the answers to all of the questions above. Some of them may not be relevant for you particular case and can be ignored. Email is generally the best way to report problems as the messages stay on a queue until they are dealt with. One of the most difficult steps in using the ***QTL Cartographer*** system is to reformat datasets. Question 2(c) above asks whether you would be willing to send us your data in order to diagnose a problem. We would like to emphasize that if you send us your data files, they will be kept in the strictest confidence. Data files sent to us are stored on a machine which cannot be accessed by the network. We will also delete your data files upon your request.

1.5.3 Contacts

For any other problems with ***QTL Cartographer*** , contact Christopher J. Basten via any of the methods listed in Table 1.3. In general, email is the best method for indicating a problem. Chris may not always get back to you right away, but will try to.

Name	Dr. Christopher J. Basten
Email	basten@statgen.ncsu.edu
Phone	(919)515-1934
Fax	(919)515-7315
Address	Program in Statistical Genetics Department of Statistics North Carolina State University Raleigh, NC 27695-8203 USA
MajorDomo	MajorDomo@statgen.ncsu.edu
Bug Report	qtlcart-bug@statgen.ncsu.edu

Table 1.3: Contact for Help

1.6 General Usage of the Programs

The programs in the ***QTL Cartographer*** suite all have the same look and feel and are heavily influenced by UNIX programs. They can be used as command line programs, or in an

interactive mode where a menu of options is presented. Some command line options that are common to all the programs are discussed in 1.6.1. The new user should become familiar with these options. In addition to the command line interface, all the programs have an interactive menu for setting options. The user need only start up any program in the suite and a list of options will appear. Selecting the number of an option will allow the user to change the value of the option. When all options are set to the user's satisfaction, choosing a zero '0' will cause the program to run. Choosing the penultimate numbered option will allow you to exit the program without changing any files. The last option saves any parameters you have set before exiting.

1.6.1 Options for all programs

Table 1.4 shows the command line parameters that are valid for all the programs in ***QTL Cartographer*** .

Working directory

A working subdirectory (folder) to hold all input and output files is a convenient way to organize your work. We suggest using a different subdirectory (folder) for each data set. In the UNIX world, you can simply change into such a subdirectory and run the programs. In the Macintosh and MS-Windows environs, you need to run the programs from where they reside and specify where the working directory is. Use the *-W* command line option to specify a working directory, or set it in the interactive menu. Be sure to follow the conventions of the particular operating system that you are working on. For UNIX, you might specify it as

```
-W /home/myaccount/qtlcart/workdir
```

While for MS-Windows it might look like

```
-W C:\qtlcart\workdir
```

And on a Macintosh, assuming that your Hard drive is called "MacintoshHD",

```
-W MacintoshHD:qtlcart:workdir
```

The programs will automatically add a file separator to the end of the path if you don't put it in. Thus

```
-W MacintoshHD:qtlcart:workdir:
```

is equivalent to the first incarnation of the Macintosh work directory. The Macintosh file separator ":" is equivalent to the DOS "\" and the UNIX "/".
You may also use relative pathnames for the working subdirectory. In the UNIX and Windows environments, a single period (.) means from here and a pair of periods (..) indicates one higher directory level. Thus,

```
-W ..\workdir
```

Option	Default	Explanation
-e	qtlcart.log	Error and Log File
-s	795793333	Random Number Seed
-h	(off)	Show help and exit
-R	qtlcart.rc	Resource File
-W	(none)	Working Directory
-A	(off)	Automatic mode
-X	qtlcart	Filename Stem
-V	(on)	Verbosity

Table 1.4: Command Line Options for all programs

would indicate go up one level from the binary subdirectory, where you will find a **workdir** subdirectory. In UNIX it might look like

```
-W ../workdir
```

For the Macintosh you use extra colons: If the binaries are in the **bin.ppc** folder inside the **qtlcart** folder, then

```
-W ::workdir:
```

would indicate that there is a folder called **workdir** in the **qtlcart**, whereas

```
-W :workdir:
```

would indicate that the **workdir** folder is inside the **bin.ppc** folder.

Listing options

Using the *-h* option will print out a list of all command line options and their values. The program will then exit without doing anything. I find this most useful when I just want a reminder of what the programs expect. This may not seem as useful now that there is an interactive menu to set options, but if you only want to use the programs in batch mode, it is a quick way to see what the values of all parameters are.

Random Number Seed

Many of the simulation programs make use of a pseudo-random number generator that requires a seed. If none is provided, the number of seconds since some date in the past is used. The *-s* option allows you to specify a seed for the random number generator. You can use this to repeat simulations to see if the same answers are obtained. If you don't use this option, the random number seed is set to the number of seconds since some arbitrary past date (for example, 1 January 1970 for Sun Workstations). The random number seed is printed to the output files of the programs on the first line. This means that if you don't specify a random number seed, each file should have a unique identifier associated with it. This identifier will also be written to the log file.

Verbosity

For debugging purposes and simply to inform the user about what is happening, many diagnostic messages will be printed out as the programs run. The user can turn these diagnostic messages off. When the messages are displayed, we refer to this as the verbosity mode. The verbosity mode can be turned off by using the the *-V* option. This means that the time and summary of options will not be printed on the standard output at runtime. This is a useful flag for batch files. Most of the messages printed to the screen are also printed to the log file.

Automatic Mode

By default, when the user starts up a program, an interactive menu for setting program options is displayed. The opposite of this is the Automatic mode. The *-A* flag turns off the interactive setting of program options. This is another flag useful for batch programming. The automatic mode should only be used by those familiar with the ***QTL Cartographer*** programs.

Resource File

A resource file is an ASCII text file that keeps track of the parameters that the user specifies in using the programs. The same file is read and updated by all the programs in the suite. You can specify a resource file using the *-R* option. It is **qtlcart.rc** by default and should be in the directory that you are currently working in (for UNIX machines) or where the binaries are (for PCs and Macintoshes). If you change any options (either via the command line or the menus), they will be saved to the file specified. If you decide to use a file other than **qtlcart.rc** as the resource file, you will need to specify it for each program you run.
Initially, the user may want to create a resource file with three lines in it. The three lines will specify the working subdirectory, online help file and a "stem" for filenames. Here is an example of a resource file for the Macintosh version of the programs:

```
-workdir      ::test:        # (The working directory)
-stem         corn           # (Stem for filenames)
-helpfile     ::qtlcart.hlp  # (The help file)
```

The working directory must be specified according to the rules of the operating system. This was explained in using the *-W* option in previous section. In the above example, a relative pathname was used. The programs will assume that there is a directory (folder) called **test** in the directory (folder) one level up from the directory (folder) that the applications reside in. The analogous lines for the MS-Windows version would look like:

```
-workdir      ..\test\       # (The working directory)
-stem         corn           # (Stem for filenames)
-helpfile     ..\qtlcart.hlp # (The help file)
```

The working directory must exist before you run ***QTL Cartographer*** .
The help file is a plain ASCII text file with indicator tokens that allow it to be used by the programs. This file is the same for all platforms and updated versions will be placed on the ftp server from time to time. The user can place the help file anywhere and indicate its placement with the "-helpfile" line in the resource file.

Filename stem

The filename stem is an important concept in the usage of ***QTL Cartographer*** . Beginning with version 1.12, the ***QTL Cartographer*** programs utilize the filename stem "qtlcart". All files are then named using this stem and filename extensions relevant to the filetype. In the resource file example above, the "-stem" entry specifies "corn" as a stem for filenames. This means that when new files are created, they will have the stem "corn" followed by a logical extension. An example would be **corn.map** for a genetic linkage map. With some practice, you will be able to know the contents of a file by its extension. You can set the filename stem on the command line with the *-X* option.

Log File

It's often useful to keep a log of the work done using the programs. The *-e* option can be used to specify the log or error file. Each time a program in the ***QTL Cartographer*** system runs, a summary of all the parameters and options is written to the log file. The file also keeps track of when the program was run and may contain other diagnostic information. The log file is appended to with each run rather than overwritten.
Remember that the log file is appended to during each invocation of any of the programs. This is something to keep in mind if you do a bootstrap in a batch file. After a thousand replications, the log file will tend to grow large. The batch file examples included with the ***QTL Cartographer*** system (see 2.4.2) take this into account by saving a copy of the log file before running the bootstrap, and deleting the large (and unnecessary) log file at the end.

Interactive Mode

The default behavior for the ***QTL Cartographer*** programs is to present the user with a menu of numbered options. This menu is in a loop, so the user can pick options and change them, one at a time. when satisfied that the proper options have been set, selecting "0" will tell the program to continue. There will always be an option to quit without doing anything. This will be the last numbered option.
When "0" is chosen, the programs will present a summary of the options and continue. At termination, the options will be written to the resource file so that the options and parameter values are remembered.
There are a few options that can only be set or changed in this interactive menu. One of these is the aforementioned filename stem which will be explained in greater detail in the next section.

One can also access the online help in the menu. There will be an option to choose help, and if the program cannot find the help file, it will ask the user for the full path and filename of the help file. The help file is an ASCII text file with tags indicating topics and subtopics. There are summaries of all the programs and their options in the help file.
A feature that is not apparent from the interactive menu is that of rewriting the resource file without doing any calculations. There is a quit command, which is the penultimate numbered command. If you choose the quit command, you will exit without rewriting the resource file. It is possible to change parameters in the menu and save them without running the program: Simply select the last value. The program will overwrite the resource file and exit without doing anything else. This is a feature for all the programs of the suite.

1.6.2 Filenaming Conventions

The ***QTL Cartographer*** system reads and creates many files and each has a default name. For example, the default output file for **Rmap** is **qtlcart.map**. We find it convenient to specify a filename stem and allow for the filename extension to indicate which program created it, and what it contains. Suppose we were working on a corn data set. We might use "corn" as the filename stem. Then Rmap would write its output to **corn.map** and its error messages to **corn.log**. Rqtl would write its output to **corn.qtl**, *etc.* Table 1.5 summarizes the standard file name extensions in the ***QTL Cartographer*** system. Beginning with version 1.12, the default behavior of ***QTL Cartographer*** is to use a filename stem: If none is given, then "qtlcart" will be the stem. Unless specifically written in the **qtlcart.rc** file, the old default names of **Rmap.out**, **Rqtl.out**, *etc.*, will no longer be used. These old default names will be used as filetype identifiers. In the output files, there will be a token "-filetype" followed by a token from the fourth column of Tables 1.5-1.6. Note that **Zmapqtl** creates some interim files, and that **Preplot** will create many other files in addition to the **GNUPLOT** control file: See Section 4.2 for details. The "-filetype" specifier will greatly aid programs such as **Rmap** and **Rcross** in translating files. As ***QTL Cartographer*** develops, this feature will be used more extensively. Once the stem is set in the menu, it will be remembered as long as a resource file is present.
In the interactive menu, if you pick an item to change (say a filename), you can wipe it out by inputting a solitary period. This way, if you had specified an input file in an earlier run, you can delete it.
In addition to the files specified in the table, we assume that files with extensions "maps" and "raw" are **MAPMAKER** genetic linkage map and raw data files, respectively. These and other files recognized by ***QTL Cartographer*** are listed in Table 1.6.

Program	Extension	Contents	-filetype
Rmap	.map	genetic linkage map	Rmap.out
Rqtl	.qtl	QTL model	Rqtl.out
Rcross	.cro	data file (markers, traits)	Rcross.out
Qstats	.qst	Qstats Analysis	Qstats.out
LRmapqtl	.lr	Single Marker Analysis	LRmapqtl.out
SRmapqtl	.sr	Stepwise Regression Analysis	SRmapqtl.out
Zmapqtl	.z	IM-CIM Results	Zmapqtl.out
JZmapqtl	.z#	Multitrait Results	JZmapqtl.out
Prune	.mpb	Pruned genetic linkage map	Rmap.out
Prune	.crb	Pruned data file	Rcross.out
Preplot	.plt	Gnuplot Control file	Preplot.plt
Eqtl	.eqt	Summary of Zmapqtl Results	Eqtl.out

Table 1.5: Standard Filename Extensions and File types for Output Files

Program	Example	Contents	-filetype
Rmap	qtlcartm.inp	genetic linkage map	map.inp
Rmap	qtlcart.maps	MAPMAKER/EXP output	mapmaker.maps
Rqtl	qtlcartq.inp	genetic model file	qtls.inp
Rcross	qtlcartc.inp	data file (markers, traits)	cross.inp
Rcross	qtlcart.raw	MAPMAKER/EXP input	mapmaker.raw
Zmapqtl	qtlcart.z3c	Perm. test interim file	ZipermC.out
Zmapqtl	qtlcart.z3e	Perm. test interim file	ZipermE.out
Zmapqtl	qtlcart.z3a	Bootstrap interim file	Ziboot.out
Eqtl	qtlcart.z3b	Bootstrap summary file	Ziboots.out
Zmapqtl	qtlcart.z3i	Jackknife interim file	Zijack.out
Eqtl	qtlcart.z3j	Jackknife summary file	Zijacks.out

Table 1.6: Miscellaneous Files and File types

Chapter 2

Simulating/Reformatting Data

The first phase in using ***QTL Cartographer*** is to create some data. You have two options for this: You can either simulate a data set or collect one yourself. The end result will be to have two files. One will contain the information on a genetic linkage map (marker order, chromosome assignment and recombination fractions) and the other a data set from a cross, which contains the markers, trait values and other explanatory variables. ***QTL Cartographer*** cannot create a genetic linkage map from a data set: You will have to use another program such as **MAPMAKER/EXP** for that task.

Figures 2.1–2.2 present a schematic of the data simulation / reformatting process. There are four main programs involved in this phase: **Rmap, Rqtl, Rcross** and **Prune**. **Rmap** is a program designed to create random genetic linkage maps, or reformat linkage maps that were prepared by **MAPMAKER/EXP**. **Rqtl** is a program that creates a genetic model for simulation. One can specify the positions, effects and the number of loci for each trait, or have the program do it randomly. Finally, **Rcross** uses the genetic linkage map and the model to create a random data set, by simulating a cross. **Rcross** can also reformat **MAPMAKER/QTL** raw data files or specially formatted data files. The fourth member of this group is **Prune**. With **Prune**, the user can eliminate individuals, markers or traits from the data set. In addition, **Prune** allows one to bootstrap or permute the data, as well as to simulate missing markers.

Regardless of whether the data are simulated or real, the important output files from this step are the genetic linkage map and the data set. We will refer to these files as **qtlcart.map** and **qtlcart.cro**, although you can name them anything you like. In fact, we generally decide on a filename stem and use filename extensions to indicate what is in the various files. If we were working on a corn data set, we might have files **corn.map** and **corn.cro** for the genetic linkage map and marker / trait data set, respectively. The naming scheme would be consistent throughout the analysis.

One note on the behavior of **Rmap**, **Rqtl** and **Rcross**. If you choose to translate a data file, then the parameters for simulations are unnecessary and they disappear from the interactive menu. If you specify no input file for any of these programs (by entering a period "." all by itself for the input filename), then the simulation parameters will reappear for the user to change.

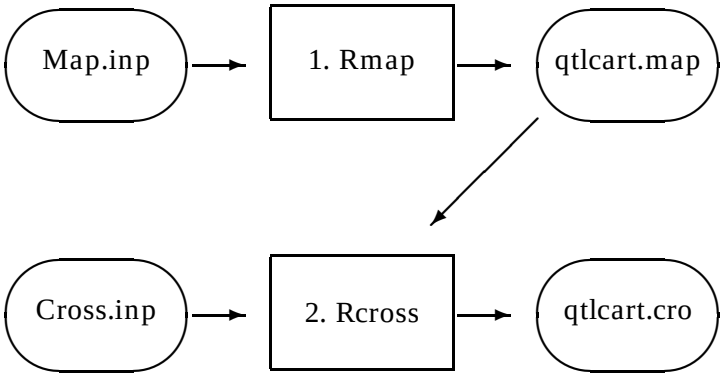

Figure 2.1: Reformatting Data

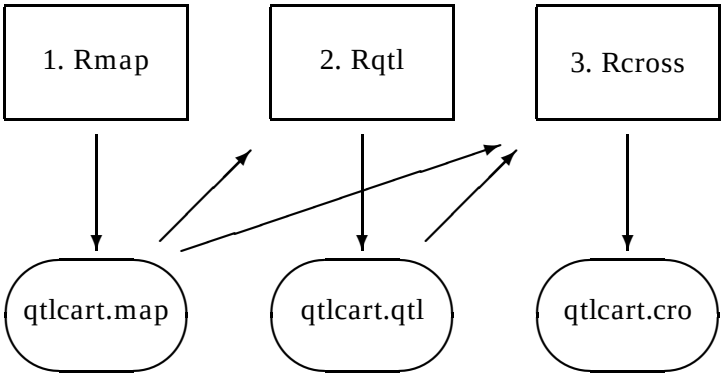

Figure 2.2: Simulating Data

2.1 Rmap

Originally, the program **Rmap** was designed to simulate a genetic linkage map. The "R" in **Rmap** was meant to convey the meaning of "Random Map". Since then we have included the ability to translate genetic linkage map information from various formats into that required by the ***QTL Cartographer*** system. Thus, the "R" can now mean reformat or random.

If you have no data, you can simulate a genetic linkage map. **Rmap** allows the user to specify the number of chromosomes, markers per chromosome and average intermarker distance for the simulation. You can also specify standard deviations for the latter two quantities. This would yield a simulated map that better approximates one that you might actually produce in the lab. Finally, you can also specify whether you want some genetic

material outside the most telomeric markers on the chromosomes.
Rmap can also read in files in three formats. The first format is the same as its output format. We will refer to this as "Rmap.out" filetype format. This feature is provided so that you can create as set of output files that **GNUPLOT** can read and display a graphic representation of your markers.
The second format is that which is produced by **MAPMAKER** (Lander et al. 1987; Lincoln et al. 1992): We will refer to it as a "mapmaker.maps" filetype format. **Rmap** will read in the **MAPMAKER** output and reformat into the "Rmap.out" format. The third format is defined in Section 6.1.2 and in the file **map.inp** included with the distribution of the programs. Remember: **Rmap** will overwrite output files. If you specify an output file that already exists, **Rmap** will destroy it when creating a new file. For this reason, we recommend that all work is done in a working subdirectory on copies of the original input files.

2.1.1 Simulating a Map

As an exercise in learning to use the programs, you can simulate a genetic linkage map. The main parameters that you will need to specify are the haploid number of chromosomes, average number of markers per chromosome, and average intermarker distance between consecutive markers. You can also simulate linkage maps in which the telomeres don't have marker information.
To see how **Rmap** simulates a genetic linkage map, denote the number of chromosomes by *c*, the average number of markers per chromosome by m and the average intermarker distance by *d* in centimorgans. Furthermore, the average amount of "tail" DNA (DNA outside the most telomeric markers) will be specified by *t*, again in centimorgans. The standard deviations of *m* and *d* will by σ_m and σ_d, respectively. All of these variables can be specified by command line options, the resource file or by the interactive menu. The standard deviation of *t* will be $\sigma_t = (\frac{t}{d})\sigma_d$. For each chromosome, **Rmap** decides how many markers are on that chromosome by picking a random number from a normal distribution with mean *m* and standard deviation σ_m. Once this is done, the amount of DNA between consecutive markers is simulated as a normal random variable with mean *d* and standard deviation σ_d. Finally, the amount of telomeric or tail DNA is simulated as a normal random variable with mean *t* and standard deviation σ_t. Setting a standard deviation equal to zero means that the quantity in question is not a random variable, but set equal to its mean value.
The parameters *c, m, d, t,* σ_m and σ_d can be set using the command line options of Table 2.1 or in the interactive menu. Note that if an input file is specified, all these parameters are ignored and **Rmap** attempts to translate the input file.
An alternate method of simulating the genetic linkage map can be invoked by changing the simulation mode parameter from 0 to 1 using the *-M* command line option. In this version, the length of the chromosomes will be normally distributed with mean *d* and standard deviation σ_d. The number of markers on a chromosome will still be normally distributed with mean *m* and standard deviation σ_m, but will be placed on the chromosome following a uniform distribution. You should set the values of d and σ_d to appropriate

levels, as they are for chromosome length rather than intermarker distance in this mode. For example, if you want roughly the same results from this mode as that in the original, then set $d = 16 \times 10 = 160$ in this mode.

2.1.2 Using MAPMAKER/EXP files

QTL Cartographer has the added capability of reading map files generated by **MAPMAKER/EXP** (Lander et al. 1987; Lincoln et al. 1992). Genetic marker order and chromosome assignment may be accomplished using **MAPMAKER/EXP**. Once map order is established, chromosomes may be saved to external files using the following **MAPMAKER/EXP** commands (in **MAPMAKER/EXP**):

```
make chromosome c1
seq M1 M2 M5 M4 M3
attach c1
framework c1
```

A chromosome c1 is defined, and the marker order (for example: M1, M2, M5, M4, M3) assigned. The "attach" and "framework" commands tell **MAPMAKER/EXP** to save this marker order on chromosome c1. See Section 5.8.1 for a more detailed example of using **MAPMAKER/EXP** to create the genetic linkage map.

After all chromosomes are defined and marker order assigned, exit **MAPMAKER/EXP**. You will find files in your directory with the extensions, "*.data", "*.maps", "*.traits", "*.xmaps". The "*.raw" file contains the original genotype and phenotype information. The "*.maps" file contains the saved marker order per assigned chromosome, as well as the estimated recombination fractions between each marker in the established order. On MS-DOS machines, the extension may be "*.map" rather than "*.maps". It would be a good idea to rename this file with a "*.mps" ending, so as not to confuse ***QTL Cartographer*** with its own genetic linkage map file.

The map order, chromosome, and recombination fraction estimate information may be used in ***QTL Cartographer*** by specifying "*.maps" as the input file for **Rmap**. The "*.raw" file is the input for the **Rcross** utility.

2.1.3 QTL Cartographer user input format

The third format is one defined for the ***QTL Cartographer*** system. It is similar to the **MAPMAKER** output format, but has commands embedded in the file to allow the program to read in the data more easily. There is an example and further explanation of this format in Section 6.1.2. It can be annotated quite freely; the example file **map.inp** is self documenting.

2.1.4 Command Line Options

Table 2.1 summarizes the command line options for **Rmap**. Most of these were explained in 2.1.1. The default options in Table 2.1 would produce a genetic linkage map on four chromosomes with 16 markers each. The markers would be equally spaced at 10 centimorgan intervals and would span the genome.

Option	Default	Explanation
-i		Input File
-o	qtlcart.map	Output File
-f	1	Map Function
-p	0.0	Map function parameter
-g	1	Output Flag
-c	4	Chromosomes
-m	16	Markers per Chromosome
-vm	0.0	Standard deviation of Markers per Chromosome
-d	10.0	Intermarker Distance (cM)
-vd	0.0	Standard deviation of Intermarker Distance
-t	0.0	Tails (Flanking DNA, in cM)
-M	0	Simulation Mode (0,1)

Table 2.1: Command Line Options for Rmap

Map Function

A map function is a mathematical relationship between recombination probabilities and map distances measured in centimorgans or Morgans. ***QTL Cartographer*** presently allows for eight map functions specified by an integer. The numbers 1, 2 or 3 correspond to the Haldane, Kosambi and Morgan (formerly Fixed) mapping functions, respectively. The default is the Haldane mapping function. If *r* corresponds to the recombination frequency between a pair of markers and d_M is the distance between them in Morgans, then the Haldane mapping function is defined by

$$d_M = -\frac{1}{2}\ln(1-2r) \tag{2.1}$$

$$r = \frac{1}{2}[1-\exp(-2d_M)] \tag{2.2}$$

The Kosambi function is

$$r = \frac{1-\exp(-4d_M)}{2[1+\exp(-4d_M)]} \tag{2.3}$$

$$d_M = \frac{1}{4}\ln[\frac{1+2r}{1-2r}] \tag{2.4}$$

and the Morgan function assumes $d_M = r$, which is complete interference. All eight mapping functions are discussed at length in Ben Lui's book (Liu 1998): We direct the reader there for the details. Table 2.2 lists the mapping functions and their integer codes for QTL Cartographer. Some of these map functions require an extra parameter. This parameter can be set in the **Rmap** menu. See Section 10.3.1 of Liu (1998) for the details.

Code	Reference	Note
1	Haldane (1919)	default
2	Kosambi (1944)	
3	Morgan (1994)	"Fixed"
4	Carter and Falconer (1951)	
5	Rao et al. (1979)	$0 \leq p \leq 1$
6	Sturt (1976)	L
7	Felsenstein (1979)	$-\infty < K < \infty, K \neq 2$
8	Karlin (1984)	binomial, $N > 0$

Table 2.2: Command Line Options for Rmap

Output Flags

The output flag takes on values of 1, 2 or 3. A 1 indicates that **Rmap** should output a file in the "Rmap.out" format. A 2 indicates that a set of files that can be plotted in **GNUPLOT** should be created while a 3 indicates that both should be done. The option to display the map in **GNUPLOT** allows a general overview of the spacing of markers. If you choose to create the **GNUPLOT** files, then **Rmap** will write one file per chromosome summarizing the linkage information. Each file will have two columns: The first indicating the position of the marker from the telomere and the second for the chromosome number. The file for chromosome 1 will be **Chrom.1**, and other files are named accordingly. Finally, a control file, **Chrom.plt**, will have the plotting commands understood by **GNUPLOT**. This file should be loaded by **GNUPLOT** to view the linkage map. Marker names are not written on the map.

Input Files

Again, note that if an input file is specified, all options from "Chromosomes" down in Table 2.1 will be ignored and **Rmap** will attempt to translate the input file. Remember that **Rmap** overwrites any files with the same name as its output file, so avoid giving your input and output files the same name.

2.2 Rqtl

Given a genetic linkage map, **Rqtl** can place a random set of quantitative trait loci on the map. The program simulates the positions and additive and dominance effects. It can also reformat a given set of QTLs defined in an input file of filetype "qtls.inp" that is explained in Section 6.2.1. The given set of QTLs might be made up by the user, or a set of estimates from a previous analysis of a data set. Table 2.3 presents the command line options for **Rqtl**. The default values from the table tell **Rqtl** to simulate nine QTLs for one trait.
For simulations, the user can specify the average number of QTLs per trait, the number of traits, and parameters for dominance and additive effects. We use the convention that Q_1 alleles are from for P_1 lines and Q_2 from P_2 lines.
Dominance can take on the values 1, 2, 3 or 4. 1 means no dominance, while 2 means Q_1 is dominant and 3 means Q_2 is dominant. A value of 4 means that dominance for each QTL will be random in magnitude and sign. The degree of dominance will be a Beta random variable *d* with shape parameters $\beta_1,\ \beta_2$. The density function for *d* is

$$f(d) = \begin{cases} \frac{d^{\beta_1-1}(1-d)^{\beta_2-1}}{B(\beta_1,\beta_2)} & \beta_1,\ \beta_2 > 0;\ -1 \le d \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

where

$$B(\beta_1,\beta_2) = \frac{\Gamma(\beta_1)\Gamma(\beta_2)}{\Gamma(\beta_1+\beta_2)} \tag{2.6}$$

and $\Gamma(x)$ is the gamma function

$$\Gamma(x) = \int_0^\infty y^{(x-1)}e^{-y}dy$$

Option	Default	Explanation
-i	(None)	Input File
-o	qtlcart.qtl	Output File
-m	qtlcart.map	Genetic Linkage Map File
-t	1	Number of Traits
-q	9	Number of QTL per Trait
-b	2.0	Additive effect parameter beta
-1	2.0	Dominance effect parameter β_1
-2	2.0	Dominance effect parameter β_2
-d	1	Dominance

Table 2.3: Command Line Options for Rqtl

The additive effects of the QTLs are independent, identically distributed random variables sampled from the gamma distribution (Zeng 1992, page 993, equation 12) and reprinted here:

$$f(a) = \frac{\beta^\beta a^{\beta-1}e^{-a\beta}}{\Gamma(\beta)}, \qquad 0 < a < \infty, \qquad 0 < \beta < \infty \tag{2.7}$$

The shape parameter β allows a wide variety of different genetic models to be generated. The additive effect of substituting an Q_1 allele for an Q_2 allele is *a*. When multiple traits are simulated, the number of QTLs per trait is simulated as a random variable with mean specified by the *-q* option.
If an input file is specified, then it is translated into a format readable by **Rcross** and the options in Table 2.3 from "Number of Traits" and below are ignored. The input file format "qtls.inp" is defined in Section 6.2.1. This input file format will allow a wide variety of genetic models to be simulated.

2.3 Rcross

Rcross uses the information generated by **Rmap** and **Rqtl** and randomly simulates a data set. Alternatively, it can also reformat **MAPMAKER** raw data files and " cross.inp" filetype formatted files. Table 2.4 presents the options for **Rcross**. The default values would create a simulated sample of 200 individuals backcrossed to P_1 with a heritability of 0.5 for the quantitative trait.

Option	Default	Explanation
-i	(None)	Input File
-o	qtlcart.cro	Output File
-m	qtlcart.map	Genetic Linkage Map File
-q	qtlcart.qtl	QTL Data File
-n	200	Sample Size
-c	1	Type of Cross
-H	0.5	Heritability
-I	0	Interactive flag
-g	0	Output format
-E	-1.000000	Environmental Variance (used if > 0)

Table 2.4: Command Line Options for Rcross

2.3.1 Simulating Data

Rcross will simulate a dataset using the genetic linkage map prepared by Rmap and the genetic model prepared by **Rqtl**. The user can specify the sample size, type of cross and heritability or environmental variance. An interactive mode allows the user to generate arbitrary crosses. **Rcross** can automatically generate backcrosses, intercrosses or any of the other experimental designs defined in Section 1.1.2. Below we describe how each individual is created. The process is repeated as many times as are necessary to get the sample size specified.

Generation of Individuals

For generating backcrosses or intercross samples, the parental lines are known. Individuals in the F_1 are all heterozygous, and all pairs of loci are in coupling. Samples derived from F_2 and later crosses need to take into account the different possible parents. This section explains how individuals are simulated in a general way.

We assume that there is one or two parental samples that will be used to create the next generation. Refer to these as lines 1 and 2. We assume monoecious, diploid individuals. To generate a new individual, one parent is selected from line 1 and one from line 2. If line 1 and line 2 are the same sample (for example, crossing two F_2 lines to form an F_3) then selfing is a possibility. Once the parents have been selected, gametes are produced, one from each parent.

The first step in producing gametes is to simulate recombination. We assume that the number of crossovers on each chromosome is distributed as a Poisson random variable with mean equal to the length of the chromosome in Morgans. A separate random integer is generated for each chromosome subject to the Poisson, and this indicates the number of crossovers on that chromosome. These crossovers are placed on the chromosome subject to a uniform distribution.

Once the crossovers are in place, gametes are generated. Starting with the first chromosome, one of the two homologs is chosen at random. This chromosome is followed until a crossover is encountered, at which point the other homolog is used. At the end of the first chromosome, a homolog from the second chromosome is chosen at random and the process continues. At the end, a gamete is created which contains the markers and QTLs. The gametes from each parent are then combined to form a new individual. Phenotypic values can then be generated.

Phenotypic Values

Phenotypic values are calculated from the genotypic values for each individual for each trait. Each individual's phenotypic value is calculated from its genotypic value with an environmental effect determined by the heritability h^2. The individual's genotypic value is based on the alleles it inherited at the quantitative trait loci. For each such locus, there will be an additive effect a that is defined in the file prepared by **Rqtl**. Genotypes will then have the following values:

Genotype	Q_1Q_1	Q_1Q_2	Q_2Q_2
Genotypic Value	$2a$	a	0

An individual's genotypic value is the sum over all loci of these values. This gives a vector of genotypic values, one entry per individual in the simulated data set. The genetic variance is the sample variance of this vector of genotypic values. Call it σ_g^2. The environmental variance, σ_e^2 is defined by

$$\sigma_e^2 = \sigma_g^2(\frac{1}{h^2} - 1) \tag{2.8}$$

where h^2 is the heritability of the trait. The extra environmental effect is taken from a normal distribution with mean 0 and variance σ_e^2. If the environmental variance is specified, the heritability is ignored and the environmental variance is used directly. For each individual in the data set, a random variable with mean zero and variance σ_e^2 is generated and added to the genotypic value. This is the phenotypic value of that individual, and is printed in the output file.

2.3.2 Translating Data

Similar to **Rmap** and **Rqtl**, **Rcross** can translate files in a pair of special formats. The first format is the input format for **MAPMAKER/QTL**. These would be the **MAPMAKER/QTL** "*.raw" files. Simply invoke **Rcross** and specify that the input file is one of these files. The parameters that are for simulations are then ignored. **REMEMBER:** The first two words of a **MAPMAKER/QTL** raw file should be "data type". Older versions of **Rcross** cannot process comments at the beginning of a raw file. In fact, it depends on those first two words to recognize the file as a **MAPMAKER/QTL** raw file. Beginning with version 1.12, comments will be allowed in the beginning of a "mapmaker.raw" file if you include the "-filetype mapmaker.raw" indicator within the first 100 lines of your file. It is usually best to put this on the first line. **Rcross** will recognize the command and translate the file appropriately. You might want to get into the habit of putting the "-filetype" token (with an appropriate identifier) in your input files, as it will become more important in future releases of ***QTL Cartographer*** . There are two other things to keep in mind when using **MAPMAKER/QTL** files. The first is that marker and trait names are truncated to eight characters in the output: Versions of ***QTL Cartographer*** prior to 1.12 will be tripped up by this. Secondly, **MAPMAKER/EXP** has been known to translate underscores "_" as minus signs "-" in its output, so you might want to avoid them.
The other format is one designed for the ***QTL Cartographer*** system. It is defined in the file **cross.inp** included with the distribution and outlined in Section 6.3.2. Finally, **Rcross** can read files in it's own output format (filetype "Rcross.out") for translation to "mapmaker.raw" or "cross.inp" filetype formats.

2.3.3 Output

The flag *-g* can be used to indicate the output format of Rcross. As with the input formats, there are three options for output. Rcross will write output in a format suitable for MAPMAKER if the *-g* option is used with the integer 2, while a cross.inp formatted file will be written with the value 1. The default is what we term the qtlcart.cro format, and is indicated by using zero with the *-g* option. Here is an example of the output of Rcross.

```
#     1472574604   -filetype Rcross.out
#
# QTL Cartographer V. 1.12c, March, 1997
#
-n        300   is the sample size
```

```
-p         63   is one more than the number of markers
-cross   B1   is the type of cross
-traits     1   is the number of traits
-Names of the traits...
  1 Trait.1
-otraits    0   is the number of other traits
#
-s
  1  1
1 1 1 1 1 1 1 2 2 2 2 2 1 1 1 1
2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 1
1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
   7.035406650635
  2  1
1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 1 1 1 1 1 1
1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1
   3.555115422473
  3  1
2 1 1 1 2 2 2 2 2 2 2 1 2 2 2 2
1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 1 1 1
2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
   4.165996548162
.
.
.
-e
```

The section prior to the '-s' token is self-explanatory. The area between the '-s' and the '-e' is the data. It starts with an identification number (1, 2, 3, *etc*), and is followed by a "1". At the moment this number "1" is ignored. With the convention that A_1 alleles originated from the P_1 line and A_2 from the P_2, marker genotypes will be encoded with the following integer values:

- 2 for A_1A_1
- 1 for A_1A_2
- 0 for A_2A_2
- 12 for A_1-, that is individuals with at least one dominant A_1 allele
- 10 for A_2-, that is individuals with at least one dominant A_2 allele

- -1 for unknown genotypes. **Rcross** read something but could not translate it.
- -2 is also for unknown genotypes. In this case, no data had been read in.

The trait values follow the marker genotypes, and finally the "other" (categorical or qualitative) traits follow at the end. The sequence repeats for all individuals. Note that there is a permissible range for trait values. By default, all trait values must be real numbers with absolute value less than one million (10^6). Any trait value that is less than negative one million is treated as a missing phenotype by the programs.

Other Traits

Other traits can be thought of as qualitative or categorical traits. Examples include sex, brood, plot, *etc.* In some cases these factors will have been "regressed out", that is a regression of the quantitative trait of interest on the categorical trait will have been performed and the residuals used as the phenotypes in the analysis. Presently, they can be input via a file of filetype "cross.inp" but not automatically analyzed. One includes these other traits in the regression model by prepending a plus sign (+) to the other trait name. For example,

```
-Names of the other traits...
  1 +Sex
  2 -Line
```

would incorporate a Sex effect in the regression model, while ignoring the Line effect.

2.4 Prune

Prune takes a genetic linkage map and a data set as input. The user can either eliminate some of the data (markers or traits), bootstrap, permute or simulate missing data. Table 2.5 summarizes the command line options for Prune.
Originally, **Prune** was strictly a command line program. In adding the interactive menu it became necessary to add a second level of interaction. When **Prune** is invoked in the interactive mode, the user will see a menu in which all the parameters of Table 2.5 can be set. The user will then proceed to another interactive menu in which data manipulation can be performed. The second menu will list actions that can be taken. The user selects an action and provides the proper values at which time the action is taken. This second menu is in a loop. The user can continue to take actions until the option to quit is chosen. At the end, the data set is printed out. A few actions can be done automatically. They are bootstrapping, permuting and simulating missing data. These are provided so that **Prune** can be run in a batch file for permutation tests or bootstrap experiments.
The output files of **Prune** may include a genetic linkage map and a data file. If markers had been eliminated, then the linkage map is regenerated to take this into account. The new output files will have the extensions .mpb and .crb, and filename stems specified by the *-o* option.

Option	Default	Explanation
-o	qtlcart	Output Filename Stem
-e	qtlcart.log	Error File
-m	qtlcart.map	Genetic Linkage Map File
-i	qtlcart.cro	Data File
-s	860436420	Random Number Seed
-I	1	Interactive mode (0,1)⇒(no,yes)
-b	0	B (1), P (2), M (3) or D (4)
-M	0.100000	Percent missing data to simulate

Table 2.5: Command Line Options for Prune

2.4.1 Pruning Datasets

The pruning of datasets occurs in an interactive menu. After setting parameters in the first menu, continue on to the second interactive menu where actions can be taken. The second interactive menu looks like this:

```
You can loop through items 1-6, but 7, 8 and 9 terminate.


 No.                Action
  1. Eliminate A- marker systems (P1 dominant)
  2. Eliminate a- marker systems (P2 dominant)
  3. Eliminate marker m on chromosome c
  4. Eliminate trait t
  5. Eliminate individuals with missing phenotypes for trait t
  6. Eliminate individuals with more than m% missing markers
  7. Bootstrap the data
  8. Permute the traits in the data
  9. Simulate m% missing markers
 10. Write modified dataset and exit
 11. Exit without writing anything
 12. Help

    Pick a number to do an action...
```

Dominant Markers

There are actions to eliminate dominant markers from the data set. These options were included at a time when **Zmapqtl** and **LRmapqtl** couldn't analyze dominant markers. With the addition of subroutines to analyze dominant markers, the need for these options has lessened. Selecting option 1 or 2 in the second interactive menu eliminates one dominant markers or one type or the other.

Eliminating markers and traits

Option 3 of the interactive menu has an action to eliminate a specific marker. You should be aware that the order of elimination is important. If all the markers to be eliminated are on separate chromosomes, the order is unimportant. If two markers from the same chromosome are to be eliminated, higher numbered marker should be eliminated first. The same concept holds for traits with option 4: Eliminate them in the order of highest to lowest. You will need to know the marker number and chromosome number rather than the marker name to use this option.

Culling sparse data

Some markers or traits may have been typed for a small proportion of individuals in the dataset. Such markers or traits can be eliminated from the data set. Option 5 will allow you to specify a trait number, and then eliminate individuals with missing data for that trait. Choosing option 6 will require a tolerance level for the percentage of missing marker data. The *-M* option specifies this number, which must be in the range $(0.0, 100.0)$. If option 6 is selected, **Prune** will eliminate individuals with this percentage of missing marker data.

Resampling data

Options 7, 8 and 9 allow the user to bootstrap, permute or simulate missing data for the dataset. If a bootstrap is chosen, then a new dataset of the same size will be resampled (with replacement) from the original data. A permutation simply permutes the trait values. The simulation of missing data requires a percentage level to simulate: That percentage of markers will then be set to unknown. These options are examined in more detail in Section 2.4.2.
Selecting 7, 8 or 9 will do the requested action, write the output and exit. The other options require you to specify when to write and exit. You also have the option of exiting without writing anything.

2.4.2 Recreating Datasets

Bootstrapping

The *-b* option with a value of 1 tells **Prune** to create a single bootstrapped data set. This option should be used alone. It will sample the data set with replacement, creating a new data set of the same sample size and writing it to the file *qtlcart.crb*. Of course, you can change the output file name by changing the output filename stem with the *-o* option. Using **Prune**, one can perform a bootstrap experiment on the data set. This is much easier to do on a UNIX workstation than a Macintosh or MS-Windows machine because it can be automated in a batch file.
For example, if one wanted to do a bootstrap experiment on a data set, one might use the sequence of commands in the following shell script, written for the C shell on a UNIX workstation.

```
#!/bin/csh
#   Bootstrap.csh
#   Usage:  Bootstrap.csh stem bootstraps email
#  where stem is the filename stem.
#        permutations is the number of permutations
#  and   email is the user's email address
#  Note:  This only works if you have set and used a filename stem,
#  and make sure that you don't use 'temp' as your stem.
#
if ( $1 == '-h' ) then
echo "    Usage:  Bootstrap.csh stem model bootstraps email"
echo "Where"
echo "          stem  = filename stem"
echo "         model  = Zmapqtl analysis model"
echo "    bootstraps  = number of bootstraps"
echo "         email  = user's email address"
echo " "
echo "Now exiting"
exit
endif
set tlog=temp.log
/usr/bin/rm -f $tlog
echo "Bootstrap experiment started " > $tlog
/usr/bin/date >>  $tlog
echo "Stem: " $1 >> $tlog
echo "Model: " $2 >> $tlog
echo "Reps: " $3 >> $tlog
echo "Email: " $4 >> $tlog
set bindir=/usr/local/bin
mv $1.log $1.logsave
set i = 1
while ( $i < $3 )
$bindir/Prune -A -V -i $1.cro -b 1  >>&  $tlog
nice $bindir/Zmapqtl -A -V -M $2 -i $1.crb  -b 1 -m $1.map >>&  $tlog
/usr/bin/mv $1.z{$2}b $1.z{$2}a
@ i++
end
mv $1.logsave $1.log
echo "Bootstrap experiment ended " >> $tlog
/usr/bin/date >>  $tlog
/usr/ucb/mail $4 <  $tlog
```

Note that the work is done in the *while .... end* loop. For each repetition, a bootstrapped data set is created with **Prune**. This data will be placed in the file ending with *.crb*. **Zmapqtl** then analyzes this bootstrapped data and updates a file with the sum and sum of squares of the test statistic and estimates of effects. You will need to have run Zmapqtl on the original data before doing the bootstrap. When this is finished, you can run **Eqtl** to get the mean and variance of the likelihood ratio, additive effect and dominance effect at each test site.

An alternate method of performing the bootstrap is similar to the above except that you omit the *-b 1* flag. In this mode, the results will be appended to the **.z* file and you would need a script to calculate the means and variances at the end of the run.

Permutation Tests

Zmapqtl can perform permutation tests using interval mapping, but if you want to do a proper permutation test using composite interval mapping and reselecting your background markers during each permutation, you will need to do it in a batch file similar to the one for bootstrapping. **Prune** can create a single permuted dataset by using the *-b* option with a value of 2. Permutation tests are then done as were the bootstrap in the previous example. Here is a UNIX shell script example for a permutation test:

```
#!/bin/csh
#   Permute.csh
#   Usage:  Permute.csh stem permutations email
#  where stem is the filename stem.
#        permutations is the number of permutations
#  and   email is the user's email address
#  Note:  This only works if you have set and used a filename stem.
#
if ( $1 == '-h' ) then
echo "    Usage:  Permute.csh stem model permutations email"
echo "Where"
echo "          stem  = filename stem"
echo "         model  = Zmapqtl Model"
echo "  permutations  = number of permutations"
echo "         email  = user's email address"
echo " "
echo "Now exiting"
exit
endif
set tlog=temp.log
/usr/bin/rm -f $tlog
echo "Permutation test started " > $tlog
/usr/bin/date >>  $tlog
echo "Stem: " $1 >> $tlog
echo "Model: " $2 >> $tlog
echo "Reps: " $3 >> $tlog
echo "Email: " $4 >> $tlog
set bindir=/usr/local/bin
set i = 1
mv $1.log $1.logsave
while ( $i < $3 )
$bindir/Prune -A -V -i $1.cro -b 2  >>&  $tlog
nice $bindir/Zmapqtl -A -V -M $2 -i $1.crb -r 1  -n 5 >>&  $tlog
@ i++
end
```

```
mv $1.logsave $1.log
/usr/bin/date >>  $tlog
/usr/ucb/mail $4 <  $tlog
```

Upon completion, you can run **Eqtl** to get the experimentwise significance thresholds. You could also have **SRmapqtl** redo the stepwise regression in the above script so that the background markers in composite interval mapping reflect the permuted data set rather than the original. There is a small quirk in this type of simulation if you are using **SRmapqtl** with stepwise forward-backward regression and **Zmapqtl** with model 6. Sometimes a permuted data set will result in no markers being sufficiently correlated with the trait of interest to be added in the forward phase of the stepwise regression. Thus, **Zmapqtl** will think there are no markers to be used as covariates, and default to interval mapping. Thus, you may not get the exact number of permutations specified to the above script.

Simulating Missing Data

You can also use **Prune** to simulate missing data. You set the amount of missing marker data you would like to simulate with the -*M* option. This will be a percent, and should be specified before you invoke the bootstrap option, which actually does the simulation. Use a value of 3 to tell **Prune** to randomly set some of the markers to missing. Over the entire data set, approximately the percentage of markers that had been set with the -*M* option will be set to -10. The results will be in a file with the filename extension ".crb". Similar to simulating missing data, some of the markers can be made dominant by using a value of 4 with the bootstrap option. The percentage of markers transformed is set with the -*M*. The direction of dominance is random: Half of those changed will convert the P_1 allele to dominant, while the other half will convert the P_2 allele.

Chapter 3

Analysis

A recent review (Doerge, Zeng, and Weir 1997) summarizes the statistical issues for mapping QTLs. It is the best place to start for a general overview of the analytical methods used in ***QTL Cartographer*** .
Figure 3.1 shows a schematic of the analysis procedure. There are five programs in this step. **Qstats** does some basic quantitative genetic statistics and summarizes missing data. It is a useful program to run at the beginning of your analysis. **LRmapqtl** does single maker analysis using linear regression. It also runs very fast and will give some idea of where QTLs are. **SRmapqtl** does stepwise regression, either forward, backward or forward with backward elimination. The final program is **Zmapqtl** which implements interval mapping (Lander and Botstein 1989) and composite interval mapping (Zeng 1993; Zeng 1994). This program generally requires more computing power. **JZmapqtl** is a module that implements multitrait mapping (Jiang and Zeng 1995).
The basic requirements for using these three programs is a genetic linkage map and a data file. The linkage map should be of filetype format "Rmap.out" and the data file of "Rcross.out". Whether the files are simulated, real or bootstrapped data is irrelevant: The analysis is the same regardless of the origin of the data.

3.1 Qstats

Qstats is a good place to start in analyzing your data. It computes some basic statistics on the quantitative traits and summarizes missing data. Let $\{y_1, y_2, \cdots, y_n\}$ be a vector of quantitative trait values. For each trait in turn, it calculates the sample size (*n*), mean ($\bar{y} = \frac{1}{n}\sum_1^n y_i$), variance ($s^2 = \frac{1}{n-1}\sum_1^n (y_i - \bar{y})^2$), standard deviation ($s = \sqrt{s^2}$), skewness, kurtosis and average deviation, $\frac{1}{n}\sum_{i=1}^n |y_i - \bar{y}|$. The coefficient of variation is the sample standard deviation divided by the sample mean.
Lynch and Walsh (1998) provide a lucid explanation of some of the statistics calculated by **Qstats**. Let the kth sample moment be $M(k) = \frac{1}{n}\sum_{i=1}^n y_i^k$. Clearly, $M(1) = \bar{y}$. Using the notation $\bar{y^k} = M(k)$, we can estimate the sample variance with

$$s^2 = \frac{n}{n-1}(\bar{y^2} - \bar{y}^2) \tag{3.1}$$

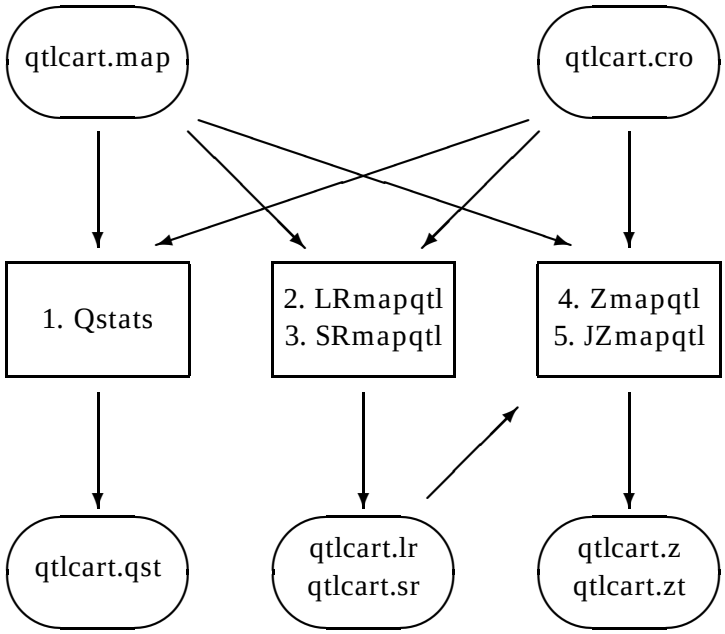

Figure 3.1: Analysis Schematic

An estimate of the skewness is

$$Skw(y) = \frac{n^2}{(n-1)(n-2)}(\bar{y^3} - 3\bar{y^2}\bar{y} + 2\bar{y}^3)$$

The standard error of skewness depends on the underlying distribution but can be approximated by $\sqrt{6/n}$. The coefficient of skewness, k_3 is

$$k_3 = \frac{Skw(y)}{s^3}$$

where the sample standard deviation, $s = \sqrt{s^2}$, is estimated from (3.1). Kurtosis is estimated by

$$Kur(y) = \frac{n^2(n+1)}{(n-1)(n-2)(n-3)}(\bar{y^4} - 4\bar{y^3}\bar{y} + 6\bar{y^2}\bar{y}^2 - 3\bar{y}^4)$$

and the coefficient of kurtosis is

$$k_4 = \frac{Kur(y) - 3s^4}{s^4}$$

Like skew, the standard error of kurtosis is dependent upon the population distribution. We give the estimate $\sqrt{24/n}$. A test of normality for the vector y then involves the test statistic

$$S = \frac{nk_3^2}{6} + \frac{nk_4^2}{24}$$

which is distributed as a χ^2 with two degrees of freedom. The critical values for the rejection of normality are 5.99 and 9.21 for tests at the 5% and 9% levels, respectively.
An example of the output follows:

```
-------------------------------------------------------
-------------------------------------------------------
This is for -trait 1 called szfreq
-------------------------------------------------------
Sample Size.............          119
M(1)....................       0.4349
M(2)....................       0.2184
M(3)....................       0.1195
M(4)....................       0.0694
Mean Trait Value........       0.4349
Variance................       0.0295
Standard Deviation......       0.1718
Coefficient of Variation...    0.3951
Average Deviation.......       0.1398
Skw..LW(24).............      -0.0010
.....Sqrt(6/n)..........       0.2245
Kur..LW(29).............       0.0022
.....Sqrt(24/n).........       0.4491
k3...LW(24).............      -0.1922
k4...LW(28).............      -0.5250
S (5%: 5.99, 1%: 9.21).....    2.0992
-------------------------------------------------------
-------------------------------------------------------
```

In the above example, LW(i) refers to a page number in Lynch and Walsh (1998) where one can find an explanation of the quantity. The value of the test statistic S is 2.0992, thus one would fail to reject the hypothesis that this trait is normally distributed.
After the basic statistics, **Qstats** draws a histogram of the quantitative trait. It is a simple histogram in that the range of the data are divided into 50 equally sized bins, and the number of data points falling into each bin are counted and plotted. A small table following the histogram gives the sample size, minimum, first quartile, median, second quartile and maximum.

3.1.1 Command Line Options

Table 3.1 summarizes the command line options for **Qstats**. There are very few of them. You can specify the data set, genetic linkage map file and output file. In addition, all the global options of Table 1.4 are valid.
Another function of **Qstats** is to summarize the missing data for markers, traits and individuals. Following the histogram, there will be a table. For each trait, it will present a summary of missing data for each marker in turn. The table will consist of seven columns.

Option	Default	Explanation
-i	qtlcart.cro	Data Input File
-o	qtlcart.qst	Output File
-m	qtlcart.map	Genetic Linkage Map File

Table 3.1: Command Line Options for Qstats

The first three columns indicate the chromosome, marker number and name of the marker (if there is a marker name). The fourth column specifies what type of marker **Qstats** thinks it is. There are three types that are recognized. The first is codominant and is indicated by a "co" token. The other two are dominant markers and **Qstats** distinguishes between marker systems in which A_1 is dominant to A_2 (indicated by the token "A-") and those in which A_2 is dominant to A_1 ("a-"). Column 5 has the counts of individuals with data for the marker, while column 6 has the counts of individuals with both marker and trait data. Column seven is just the ratio of columns 5 and 6.
At the end of the **Qstats** output file, there will be a summary of missing data for each individual in the data set. **Qstats** will indicate the number of marker systems, quantitative traits and categorical traits. It will then have a table with seven columns. Column 1 is for the individual. Column 2 indicates the number of markers for which the individual is typed, and Column 3 indicates a percent. Columns 4 and 5 do the same for traits while columns 6 and 7 summarize the data for categorical traits.
Something to keep in mind is that some of the analyses require large sample sizes. For example, if the sample sizes are too small, the ECM algorthm may fail in **Zmapqtl**. When difficulties in analysis are encountered, check the missing data summaries in the **Qstats** output: Such problems often correspond to areas with a lot of missing data.

3.1.2 Segregation

Qstats also tests for adherence to Mendelian segregation at all marker loci. For a given locus, suppose there are r genotypic classes. Let p_i be the expected frequency, and n_i the observed count for the ith class. For a sample of size n, the expected counts will be np_i and the observed frequencies will be n_i/n. We can construct a test statistics based on a contigency table

$$T_1 = \sum_{i=1}^{r} \frac{(n_i - np_i)^2}{np_i} = n \sum_{i=1}^{r} \frac{(n_i/n - p_i)^2}{p_i}$$

or a comparison of likelihoods

$$T_2 = -2 \sum_{i=1}^{r} n_i (\ln n_i - \ln np_i)$$

Both T_1 and T_2 should have a χ^2 distribution with one degree of freedom. Both statistics are calculated and presented in a table in the **Qstats** output.

3.2 LRmapqtl

LRmapqtl fits the data to a simple linear regression model. For each marker in turn, it fits a simple linear model to the trait data. It is a quick way to get an idea of where the QTLs may reside.

3.2.1 Simple Linear Regression

For each marker in turn, **LRmapqtl** fits the phenotypic data to the linear model

$$y_i = b_0 + b_1 x_i + e \tag{3.2}$$

where y_i is the phenotype of the ith individual and x_i is an indicator variable for the marker genotype. Generally,

$$x_i = \begin{cases} 2 & \text{if} \quad A_1A_1 \\ 1 & \text{if} \quad A_1A_2 \\ 0 & \text{if} \quad A_2A_2 \end{cases}$$

but for B_1 crosses

$$x_i = \begin{cases} 1 & \text{if} \quad A_1A_1 \\ 0 & \text{if} \quad A_1A_2 \end{cases}$$

If the marker is missing or dominant, then an expected value for the marker is calculated from the flanking markers (Fisch, Ragot, and Gay 1996; Jiang and Zeng 1997). The regression parameters b_0 and b_1 can be estimated, and e is assumed to have a normal distribution. **LRmapqtl** can also take into account categorical traits, that is other variables such as sex or brood, in its analysis. If your data set contains such information, then there should be a list of the names of these other variables near the beginning of the "Rcross.out" formatted file. These names might look as follows:

```
-Names of the other traits...
  1 Sex
  2 Line
```

If you would like to include "Sex" and "Sex by Marker interaction" terms in your analysis, then you need to indicate as much to **LRmapqtl**. If you prefix the name of one of these variables with a plus sign $(+)$, then it will be incorporated into the linear model.

```
-Names of the other traits...
  1 +Sex
  2 Line
```

In **LRmapqtl**, this would consider both Sex and Sex by Marker interaction terms. In **Zmapqtl** and **SRmapqtl**, the Sex by Marker term wouldn't be incorporated, but the Sex factor would. All other variables that have no + sign at the beginning of their names will be ignored in the analysis. For the above example, a pair of models will be considered:

$$y_i = b_0 + b_1 x_i + b_2 Sex + b_3 Sex \times x_i + e \tag{3.3}$$

$$y_i = b_0 + b_2 Sex + e \tag{3.4}$$

Option	Default	Explanation
-i	qtlcart.cro	Data Input File
-o	qtlcart.lr	Output File
-m	qtlcart.map	Genetic Linkage Map File
-r	0	Number of permutations
-t	1	Trait to analyze

Table 3.2: Command Line Options for LRmapqtl

The output will give probabilities that the marker is significant.

Table 3.2 shows the command line options specific to LRmapqtl. As with **Qstats**, there are few parameters to change. The *-t* option allows you to specify a trait to analyze. It is trait 1 by default. If you only have one trait, you can ignore this option. If your data set has more than one trait, you can analyze a specific trait by using *-t* with an integer from 1 to the number of traits. If you want LRmapqtl to analyze all traits, use a value greater than the number of traits.

3.2.2 Output

LRmapqtl prints out a histogram of the trait (identical to the one from **Qstats**), and the results of simple linear regression. The results are displayed in a table with seven columns. The first column indicates the chromosome, while the second gives the number of the marker on the chromosome. The name of the marker can be found in the genetic linkage map file. The next two columns correspond to the parameters in the linear model (Equation 3.2). Column three is the intercept and column four the slope of the least squares regression line fit to the data. Column five is a likelihood ratio test statistic for the model, and column six is the *F* statistic. Column seven is the tail probability of the *F* statistic assuming one and $n-1$ degrees of freedom in the numerator and denominator, respectively. Asterisks attached to these probabilities indicate significance of the *F* statistics: Significance at the 5%, 1%, 0.1% and 0.01% levels are indicated by one, two, three and four asterisks, respectively.

The results of running **LRmapqtl** are used in **Zmapqtl** for analysis models four and five (see Section 3.4.2).

3.2.3 Permutation Tests

The *-r* option tells **LRmapqtl** to perform a permutation test (Churchill and Doerge 1994). The argument to *-r* indicates how many permutations should be performed. In each permutation, the phenotypes are shuffled relative to the genotypes over individuals and the analysis is redone. The results are summarized at the end of the **LRmapqtl** output file.

3.3 SRmapqtl

SRmapqtl uses the technique of stepwise regression to search for QTLs. For forward and backward regression, it simply ranks the markers for their effect on the quantitative trait. In forward stepwise regression (FS), each marker in turn is tested for its effect on the quantitative trait using linear regression. That marker with the largest partial F-statistic is assigned rank 1 and included in all subsequent analyses. Step two tests all the remaining markers, and assigns rank 2 to the marker with the largest partial F-statistic. This is repeated until all the markers have been ranked.

Option	Default	Explanation
-i	qtlcart.cro	Input File
-o	qtlcart.sr	Output File
-e	qtlcart.log	Error File
-m	qtlcart.map	Genetic Linkage Map File
-s	860437285	Random Number Seed
-M	0	FS, BE or FB (0,1,2)?
-t	1	Trait to analyze
-F	0.1	Size: p(Fin) =
-B	0.1	Size: p(Fout) =

Table 3.3: Command Line Options for SRmapqtl

Backward elimination regression (BE) starts with all markers in the model. In the first step, each marker in turn is removed and a partial F-statistic is calculated. That marker with the smallest partial F statistic is given the lowest rank and removed from subsequent analyses. This is repeated until all the markers have been ranked.
The above methods seek only to rank the markers: They make no effort to determine whether adding or deleting a marker makes a significant difference for the fit of the model to the data. A third method (FB) is to start with forward stepwise regression, but only keep adding markers while the p-value of the partial F statistic of the marker to be added is below a defined threshold, $p(F_{in})$. When a step is reached in which no more markers can be added, all of the markers are retested to see if they are still significant. Each marker in turn is deleted from the model, a p-value is calculated for the partial F-statistic, and if the p-value is greater than a specified level $p(F_{out})$, it is deleted.
As with **LRmapqtl**, any otraits that begin with a plus sign are also used in the regression model. Unlike **LRmapqtl**, no interaction terms are used. The command line parameters for **SRmapqtl** are listed in Table 3.3. One added feature is that if you use the *-t* option with an integer value one greater than the number of traits, then all traits will be analyzed in turn.

3.3.1 Output

For the specified trait, **SRmapqtl** will output a small table:

```
-----------------------------------------------------
-----------------------------------------------------
Chromosome    Marker      Rank         F-Stat      DOF
-----------------------------------------------------  -start
         1         1         2       13.38778       114
         2         3         4       10.12742       110
         3         1         5        3.55528       108
         3         2         3       11.15490       112
         4         3         1       28.85236       116  -end
-----------------------------------------------------
-----------------------------------------------------
```

The first two columns indicate the chromosome and marker. The third column gives the rank of that marker as determined by the stepwise regression mode of choice. Then there will be an F-statistic indicating the difference between having that variable in the model or not. Finally, the DOF (degrees of freedom) for the numerator of that F statistic is given. For forward stepwise or backward elimination, **SRmapqtl** will try to rank all of the markers no matter how small the F statistic is. For the forward regression with backward elimination, the program proceeds to add variables until the F statistic p-value is less than that specified by the *-F* option (0.1 by default). Then **SRmapqtl** rechecks all the variables added and will eliminate any with an F statistic p-value less than the value given with the *-B* option.

In general, the FB method is probably the best method for picking background markers to be used with model 6 in **Zmapqtl** and **JZmapqtl**. To this end, **SRmapqtl** should be run prior to using either module. **Zmapqtl** and **JZmapqtl** will read the results of **SRmapqtl** and use the markers that are ranked. You can specify an upper bound to the number of background parameters to be used in **Zmapqtl**. **JZmapqtl** will use all the markers that are listed for all traits in its analysis: The FB method thus selects only a subset of significant markers.

Be aware that **SRmapqtl** tries to determine how many markers can be analyzed at once. The number of parameters has to be smaller than the sample size. If you try to use backward regression, and there are more markers than individuals, then **SRmapqtl** will default to forward stepwise regression and rank as many markers as possible. You should be aware that when dominance can be estimated, each marker will count two towards the total number of parameters and you will need a sample size of at least twice the number of markers to do backward elimination.

3.4 Zmapqtl

Zmapqtl implements interval and composite interval mapping. There are also options to perform a permutation test (Churchill and Doerge 1994; Doerge and Churchill 1996).

3.4.1 Computational Methodology

Composite interval mapping (Zeng 1993; Zeng 1994) combines interval mapping with multiple regression. The statistical model is defined as

$$\mathbf{Y} = \mathbf{x}^* b^* + \mathbf{z}^* d^* + \mathbf{XB} + \mathbf{E} \tag{3.5}$$

where

- $\mathbf{Y}$ is a vector of trait values
- b^* and d^* are the additive and dominance effects of the putative QTL being tested
- $\mathbf{x}^*$ and $\mathbf{z}^*$ are indicator variable vectors specifying the probabilities of an individual being in different genotypes for the putative QTL constructed by flanking makers
- $\mathbf{B}$ is the vector of effects of other selected markers fitted in the model
- $\mathbf{X}$ is the marker information matrix for those selected markers
- $\mathbf{E}$ is the error vector.

Estimates of the parameters are obtained by maximum likelihood through an ECM (for Expectation / Conditional Maximization) algorithm (Meng and Rubin 1993). In each E-step, the probability of an individual being in different genotypes of the putative QTL is updated. In the CM-step, the estimation of parameters b^* and d^* is separated from that of $\mathbf{B}$, and each group is estimated conditional on the others. This procedure is implemented for numerical consideration. As $\mathbf{x}^*$ and $\mathbf{z}^*$ are separated from $\mathbf{X}$, $\mathbf{X}$ is unchanged in each iteration, and its costly recalculation is avoided.
For an F_2 population, the hypotheses for testing are $H_0 : b^* = 0 \text{ and } d^* = 0$ and $H_3 : b^* \neq 0 \text{ or } d^* \neq 0$. This is performed through a likelihood ratio test procedure. In addition, it is possible to test hypotheses on b^* and d^* individually. For a backcross data set, dominance cannot be estimated and d^* is dropped from Equation 3.5.
The trait will have a variance s^2. Under the null hypothesis

$$H_0 : Y = XB + E$$

the sample variance of the residuals will be s_0^2. For a given alternative model, say

$$H_1 : \mathbf{Y} = \mathbf{x}^* b^* + \mathbf{z}^* d^* + \mathbf{XB} + \mathbf{E}$$

the variance of the residuals would be s_1^2. With this in mind we can calculate the proportion of variance explained by a QTL at the test site. The quantity is usually called r^2 and estimated by

$$r^2 = \frac{s_0^2 - s_1^2}{s^2}$$

An alternative estimate would use the total variance. Denote it by

$$r_t^2 = \frac{s^2 - s_1^2}{s^2}$$

r^2 is the proportion of the variance explained by the QTL conditioned on the background markers and any explanatory variables. r_t^2 is the proportion of the total variance explained by the QTL and the the background markers and any explanatory variables. Generally, $r_t^2 \geq r^2$

3.4.2 Models

When we speak of models for analysis, we mean to specify the markers used as cofactors in composite interval mapping. There are presently six models for analysis.

1. Use all the markers to control for the genetic background. This is model 1 from Zeng (1994).

2. Use all unlinked markers to control for the genetic background. This is model 2 from Zeng (1994).

3. Don't use any markers to control for the genetic background. This is also known as interval mapping and is the same as Lander and Botstein's method (Lander and Botstein 1989).

4. This is an ad-hoc model. One marker from each chromosome (except for the chromosome on which we are testing) is used to control for the genetic background. The results of **LRmapqtl** are scanned, and the marker that showed the highest test statistic from each chromosome is used.

5. This is another ad-hoc model. Two markers from each chromosome are used to control for the genetic background. They are the top two markers as determined by **LRmapqtl**. In addition, all the other markers on the chromosome of the test position that are more than 10 cM away from the flanking markers are also thrown in. It may be ad-hoc, but tends to work best at this time. The value of 10 centimorgans can be changed with the *-w* option.

6. Model six will be explained in the next subsection.

7. Model seven requires the results of a prior run of **Zmapqtl** and **Eqtl**. Initially, the user may want to run **Zmapqtl** with interval mapping, summarize the positions and effects of that analysis using **Eqtl**, and then use those estimates as the covariates in the regression model. Virtual markers are created at the best estimates for the positions of the QTLs.

Zmapqtl Model Six

Model 6 requires two additional parameters. One is the number of markers to control for the genetic background (n_p), and the other is a window size (w_s). When invoked, the program will read in the results of a prior run of **SRmapqtl** to pick the most important markers to control for the genetic background. Then, when testing at any point on the genome, it will use up to n_p of these markers. If **SRmapqtl** didn't rank as many markers

as specified with n_p, then n_p is reset to the number of markers ranked. The window size will block out a region of the genome on either side of the markers flanking the test site. Since these flanking regions are tightly linked to the testing site, if we were to use them as background markers we would then be eliminating the signal from the test site itself.
Note that if $w_s = 0.0$ and n_p equals the total number of markers, then Model 6 reduces to Model 1. If w_s is large (say the size of the largest chromosome) and n_p equals the number of markers, then Model 2 is the result. If n_p is zero, then Model 3 is the result. In the future, we will recommend that people use model 3 or model 6 for analysis. The default values of 5 for n_p and 10 for w_s should be good starting points for Model 6. Increasing n_p will allow better resolution for mapping linked QTLs.

3.4.3 Zmapqtl Options

Table 3.4 shows the command line options specific to **Zmapqtl**. One can select a trait to analyze, a model for analysis and a walking speed along the genome (that is, the interval between successive analysis points). The user can analyze just one chromosome or the entire genome. Finally, permutation tests or bootstraps can be performed by setting the number of permutations or bootstraps to a number greater than 0. Explanatory variables such as Sex or Line are automatically included in the analysis if their names are preceded by a plus sign in the data file. This is similar to **LRmapqtl**, except that interaction terms are not yet used.

Option	Default	Explanation
-i	qtlcart.cro	Input File
-o	qtlcart.z	Output File
-m	qtlcart.map	Genetic Linkage Map File
-l	qtlcart.lr	LRmapqtl Results file
-S	qtlcart.sr	SRmapqtl Results file (Model 6)
-M	3	Model
-t	1	Trait to analyze
-c	0	Chromosome to analyze
-d	2.0	Walking speed in cM
-n	5	Number of Background Parameters (Model 6)
-w	10.0	Window Size in cM (Models 5 and 6)
-r	0	Number of Permutations
-b	0	Number of Bootstraps

Table 3.4: Command Line Options for Zmapqtl

Traits and Chromosomes

The *-t* option allows the user to specify which trait in a data set with multiple traits is to be analyzed. For multiple trait analysis, use **JZmapqtl**. If you set the trait number to one

more than the total number of traits, then all traits (except for those whose names begin with a minus sign) will be analyzed in succession: This only works with models 1, 2, 3 and 6.
One can also limit the analysis to a single chromosome with the *-c* option.

Background Parameters and Window Sizes

For models 5 and 6, one can specify the size of the window (w_s) on either side of the test interval that is blocked from having markers in the background. This option is ignored for all models except 5 and 6. The number of background parameters (n_p) is only used with model 6 and is explained above.

Permutations, Bootstraps and Jackknives

Zmapqtl allows for permutation tests and bootstrap or jackknife resamplings. The former is a way to determine experimentwise significance levels and comparisonwise probabilities (Churchill and Doerge 1994; Doerge and Churchill 1996). Phenotypes are shuffled against genotypes and the analyses are redone. For each test position, the comparisonwise probability or *P* value is the proportion of permuted datasets that have test statistics less than the observed data set test statistic. It should correspond to the probability of the observed test statistic assuming a χ^2 distribution with one degree of freedom. For the experimentwise significance level, the highest test statistic in each permutation is recorded, and these are ordered at the end of the permutations. The 90, 95, 97.5 and 99th percentile values are then the experimentwise significance levels at $\alpha = 0.1,\ 0.05,\ 0.025$ and 0.01, respectively. Permutation tests are done for interval mapping within **Zmapqtl**, and interim results are stored in the files qtlcart.z3c and qtlcart.z3e. There are two distinct ways to perform the permutation test in ***QTL Cartographer*** . The first is simply to have **Zmapqtl** do the permuting and analysis: You would then use *-r* with the number of permutations to perform. If you choose to do the permutation test entirely within **Zmapqtl**, you must set the number permutations to a value larger than number of permutations already completed. In this way, if you started a permutation test and your machine crashed before the test was complete, you can restart **Zmapqtl** and finish it from where it left off.
An alternative way to do the permutation test is in a batch file. For composite interval mapping, one might want to reselect the background markers with **SRmapqtl** in each permutation. To this end, one would need to permute the traits, reselect the background markers and then run the composite interval mapping. The shell script example in Section 2.4.2 shows how to do this. Since **Prune** has already permuted the traits, we want **Zmapqtl** to read in the data, do the analysis without permuting the traits, and write the interim results to the appropriate files. Setting the number of permutations equal to one is a special indicator to **Zmapqtl** to do just that.
In the bootstrap, new datasets are created from the original by sampling with replacement. New datasets are the same size as the original. The statistics are redone and printed out. See the section **Prune** as to how to do bootstrapping.

Interim file	Created during	Contains
qtlcart.z6e	permutation test	Experimentwise state
qtlcart.z6c	permutation test	Comparisonwise state
qtlcart.z6a	bootstrap resampling	Iteration i bootstrap
qtlcart.z6b	bootstrap resampling	Iteration $i+1$ bootstrap
qtlcart.z6i	jackknife resampling	Iteration i jackknife
qtlcart.z6j	jackknife resampling	Iteration $i+1$ jackknife

Table 3.5: Examples of Interim Files for Model 6

Jackknife resampling is performed by calculating n (the sample size) new estimates of the parameters: The ith estimate is calculated by deleting individual i from the dataset. The standard deviation over these n new estimates provides an estimate of the standard deviation for the test statistic and additive and dominance effects. You invoke the Jackknife by setting the number of bootstraps to 2. **Zmapqtl** uses two interim files to perform the jackknife. If you are using Model 6 in **Zmapqtl** and your filename stem is qtlcart, then these files will be called qtlcart.z6i and qtlcart.z6j. These files contain the sum and sum of squares up to the previous and current iteration, as **Zmapqtl** runs. Initially, the qtlcart.z6i file contains columns of zeros: This is the sum before any iterations are performed. Subsequently, qtlcart.z6j will contain the interim state after each odd-numbered iteration, while qtlcart.z6i will contain the state after each even-numbered iteration. If individual i has no trait data, then the ith iteration will be skipped. For this reason, one cannot be sure that the file ending in "j" is the last iteration for odd sample sizes. It is best to look at both files at the conclusion of a jackknife experiment, and rename the interim file with the greater number of iterations to qtlcart.z6i. It this is done, then **Eqtl** will recognize it and calculate the means and sample standard deviations of the test statistic and effects.
To clarify the interim file names, we consider an example using Model 6 in **Zmapqtl** and the default filename stem "qtlcart". Table 3.5 lists the interim file names. **Eqtl** automatically looks for files named "qtlcart.z6e", "qtlcart.z6a" and "qtlcart.z6i". These files will be processed and the appropriate calculations done. **Eqtl** will overwrite the "qtlcart.z6b" and "qtlcart.z6j" files after completing its calculations, so if you want to save them, do so before running **Eqtl**. If you chose to use another model (say model 3), then the "6" in the filenames of Table 3.5 would be a "3".

3.4.4 Output

Here is a truncated example of the output of **Zmapqtl** for a backcross.

```
#      890840384   -filetype Zmapqtl.out
#
#       QTL Cartographer V. 1.13b, March 1998
#       This output file (qtlcart.z) was created by Zmapqtl...
#
```

```
#       It is 10:39:44 on Wednesday, 25 March 1998
#
#
#The position is from the left telomere on the chromosome
-window      10.00      Window size for models 5 and 6
-background      5      Background parameters in model 6
-Model           6      Model number
-trait           1      Analyzed trait [Trait_1]
-cross          B2      Cross
#  Test Site   * Like. Ratio Test Statistics   *     Additive
 c  m position     H0:H1      R2(0:1)    TR2(0:1)    H1:a        S1
-s
 1  1  0.0001      0.411      0.002      0.473      0.027      1.531
 1  2  0.0133      0.016      0.000      0.472      0.005      1.542
 1  2  0.0333      0.023      0.000      0.472      0.006      1.547
 1  2  0.0533      0.031      0.000      0.472      0.008      1.554
 1  2  0.0733      0.041      0.000      0.472      0.009      1.563
 1  2  0.0933      0.052      0.000      0.472      0.010      1.572
 1  2  0.1133      0.063      0.000      0.472      0.011      1.582
 1  2  0.1333      0.073      0.000      0.472      0.012      1.593
.
.
.
-e
```

For a backcross, let a be the additive effect. We have two hypotheses:

- H_0: no QTL effect at the test position, *i.e.* $a = 0$
- H_1: There is a QTL effect at the test position, *i.e.* $a \neq 0$

The first eight columns correspond to

1. Chromosome of test position
2. Left flanking marker of test position
3. Absolute position from left telomere, in Morgans.
4. Likelihood ratio test statistic for $\frac{H_1}{H_0}$. It is a χ^2 random variable with one degree of freedom for any position, meaning that a value of 3.84 or higher is evidence for a QTL. The significance level over more positions will be higher due to multiple testing.
5. r^2
6. r_t^2
7. Estimate of a (the additive effect) under H_1
8. Test statistic S for the normality of the residuals under H_1

The last 13 columns are not shown because they are only valid for F_2 design experiments. They would all be zeros if shown.
The output for an F_2 design (or any design in which dominance effects can be estimated) is similar, but has more information. For an F_2, you can estimate additive (*a*) and dominance (*d*) parameters at each position. Thus, there are four hypotheses.

- H_0: $a = 0$, $d = 0$
- H_1: $a \neq 0$, $d = 0$
- H_2: $a = 0$, $d \neq 0$
- H_3: $a \neq 0$, $d \neq 0$

and twelve full columns of output, corresponding to all possible hypothesis tests and parameter estimates. The 21 columns correspond to

1. Chromosome of test position.
2. Left flanking marker of test position.
3. Absolute position from left telomere, in Morgans.
4. Likelihood ratio test statistic for $\frac{H_3}{H_0}$.
5. Likelihood ratio test statistic for $\frac{H_3}{H_1}$.
6. Likelihood ratio test statistic for $\frac{H_3}{H_2}$.
7. Estimate of a (the additive effect) under H_1.
8. Estimate of a (the additive effect) under H_3.
9. Estimate of d (the dominance effect) under H_2.
10. Estimate of d (the dominance effect) under H_3.
11. Likelihood ratio test statistic for $\frac{H_1}{H_0}$.
12. Likelihood ratio test statistic for $\frac{H_2}{H_0}$.
13. r^2 for $\frac{H_1}{H_0}$.
14. r^2 for $\frac{H_2}{H_0}$.
15. r^2 for $\frac{H_3}{H_0}$.
16. r_t^2 for $\frac{H_1}{H_0}$.
17. r_t^2 for $\frac{H_2}{H_0}$.
18. r_t^2 for $\frac{H_3}{H_0}$.
19. S for H_1.
20. S for H_2.
21. S for H_3.

Permutation Test output

If you chose to do a permutation test (Churchill and Doerge 1994) for the purpose of estimating experiment specific threshold values, **Zmapqtl** will create two auxiliary files to store interim comparisonwise and experimentwise test statistics. If the filename stem is "qtlcart" and the model for analysis is "6", then these files will be "qtlcart.z6c" and "qtlcart.z6e". The former file should look something like this:

```
#Row Chrom Mark Position  Original    P-Val     Count    -perm 899
   -start
    1   1   1  0.00010    0.00000   0.982202        883
    2   1   1  0.02010    0.00000   0.976641        878
.
.
.
```

whose columns are

1. Integer indicating the row.
2. Chromosome of test position.
3. Left flanking marker of test position.
4. Absolute position of test from left telomere, in Morgans.
5. Likelihood ratio test statistic of actual data. For backcrosses, this is $\frac{H_1}{H_0}$, while for F_2's, it is $\frac{H_3}{H_0}$.
6. Proportion of permuted data sets with an LR greater than or equal to the observed LR.
7. Actual count of the number of permuted data sets with an LR greater than or equal to the observed LR.

In each step of the permutation test, this file is rewritten and the number following the "-perm" token incremented. This way, if the computer crashes during a run, **Zmapqtl** can be restarted from where it left off. If you were running **Zmapqtl** with 1,000 permutations, and the process stopped at 899 as above, then restarting **Zmapqtl** with 1,000 permutations will begin with permutation 900 and continue to 1,000.
The second file, "qtlcart.z6e", will contain two columns of numbers: the permutation and the maximal likelihood ratio over the genome in that permutation. Each permutation will add a line to the output. When enough permutations have been done, **Eqtl** can be run to summarize the experimentwise levels. A small table will be written to the log file that looks like:

```
-start
 Performed 899 permutations of the phenotypes and genotypes
Here are the Experimentwise significance levels for different alpha
Permutation significance level for alpha = 0.1   : 11.6858
Permutation significance level for alpha = 0.05  : 13.3108
```

```
Permutation significance level for alpha = 0.025 : 14.6669
Permutation significance level for alpha = 0.01  : 16.8008
-end of shuffling results
```

For each shuffle, the largest likelihood ratio test statistic over all test positions is saved in the file. At the end of the shuffling, these maximum values are sorted, and the $(1-\alpha) \times 999$ th largest is the experimentwise significance level for a test of size α. The number of permutations can be changed from 899 to any integer from 0 to 10,000. This upper bound could be made higher by changing the appropriate definition in the Main.h source file and recompiling. In general, we find that 1000 permutations is a sufficient number. In a test, values of 1000 and 17,000 were used with little difference in the ultimate comparisonwise and experimentwise values.

3.5 JZmapqtl

JZmapqtl implements interval and composite interval mapping for multiple traits (Jiang and Zeng 1995). It is very similar to **Zmapqtl** except that it can jointly analyze more than one trait. It is best used after **Zmapqtl** when one suspects that two traits are correlated.

3.5.1 JZmapqtl Options

Table 3.6 shows the command line options specific to **JZmapqtl**. Most are the same as those for **Zmapqtl**. One thing to note is that there is no facility for permuation tests or bootstraps at this time.

Option	Default	Explanation
-i	qtlcart.cro	Input File
-o	qtlcart.z	Output File
-e	qtlcart.log	Error File
-m	qtlcart.map	Genetic Linkage Map File
-S	qtlcart.sr	SRmapqtl results (Model 6)
-E	qtlcart.eqt	Eqtl results (Model 7)
-s	893339277	Random Number Seed
-M	3	Model [3,6,7], 3=>IM
-t	1	Trait to analyze
-c	0	Chromosome to analyze (0=>all)
-d	2.000000	Walking speed in cM
-n	5	Number of Background Parameters (Model 6)
-w	10.000000	Window Size in cM (Model 6)
-I	1	Hypothesis test

Table 3.6: Command Line Options for JZmapqtl

3.5.2 Output

JZmapqtl will create a number of different output files depending on the number of traits in the joint analysis. There will be one file per trait that has estimates for the parameters for that trait. These files will end in .z#, where # is a number indicating the trait. There will be one other file, ending in .z0 that contains the results of the joint liklihood ratio.
The joint results file ending in .z0 will have four columns corresponding to the chromosome, marker, markername and test position. Then there will be column giving the joint liklihoods for the test position for all possible hypothesis tests (see next section).
The single trait files, ending in .z#, will have the results for the numbered trait. In addition to the chromosome, marker, markername and test position, the likelihood ratio and parameter estimates will be given. All columns are labelled, and the parameters are the same as explained in the **Zmapqtl** section.

3.5.3 Usage Hints

Trait Selection

You can select traits to include in the analysis in three ways. Suppose that you have t traits in your data file.

1. Set the trait to analyze at 0, so that no traits except those beginning with a $+$ (plus sign) are analyzed. You would need to edit the .cro file first to prepend a $+$ to all traits you want in the analysis.

2. Set the trait to a value in the range $1 - t$, inclusive. You will then get single trait results for the selected trait.

3. Set the trait to a value greater than t. All traits will be put in the analysis, unless they begin with a $-$ (minus sign).

Hypothesis tests

You need to set the hypothesis test for SF_x and RF_x crosses. The default of 1 is fine for crosses in which there are only two marker genotypic classes (backcrosses and recombinant inbreds). For SF_x and RF_x, values of 30, 31 or 32 are valid. Recall that we have the following hypotheses:

1. $H_0 : a = d = 0$

2. $H_1 : a \neq 0, d = 0$

3. $H_2 : a = 0, d \neq 0$

4. $H_3 : a \neq 0, d \neq 0$

For 30, we test $H_3 : H_0$. For 31, we test $H_3 : H_0$, $H_3 : H_1$ and $H_1 : H_0$. For 32, we test $H_3 : H_0$, $H_3 : H_2$ and $H_2 : H_0$. 30 is probably fine for initial scans.
Also, if you do only have two genotypic classes, then 10 is the same as 1 for the hypothesis test.

Model 6

For Model 6, be sure to run **SRmapqtl** first. Once done, **JZmapqtl** will use all markers that are significant for any of the traits in the analysis. We need to work out a better way to select the cofactors. Now it uses any markers that are significant for any trait. Also, be sure to use FB regression, or else you will end up using all markers as cofactors.

G x E Analysis

One special case of G x E analysis has been incorporated into **JZmapqtl**, namely the situation where a set of genotypes is raised in more than one environment. The value of the trait in each environment is treated as a separate trait for the common genotype. For this type of data, use hypothesis 14 or 34 to invoke the G x E analysis. Hypothesis 14 is for data with two marker genotypes, while 34 is for three marker genotypes. There will be an extra column in the output that give a likelihood ratio for a G x E effect versus no effect. When running Eqtl subsequent to doing a G x E analysis, be sure to specify the same hypothesis test.

Chapter 4

Visualization of Results

The final step in analyzing your data will be to summarize your results, either graphically or as a compact set of estimates for QTL positions and effects. We have provided some utilities that read the output of the analysis programs and reformat it for use in graphics packages. The freeware program **GNUPLOT** is recommended as a graphics engine, but the results could be plotted in any plotting package on any machine. All of the results from the analysis programs are simple text files, and all the reformatted files are also simple text. Figure 4.1 is a schematic of the programs and files that are involved in this step. **Eqtl** is a utility that quickly picks out the possible QTLs from the results of **Zmapqtl**. **Preplot** can read the output of **Rqtl**, **LRmapqtl** and **Zmapqtl** and produce simple files containing two columns of text corresponding to the values for the abscissa and ordinate of a plot. These files in turn can be plotted by **GNUPLOT**, or imported into various plotting packages on various platforms.

4.1 Eqtl

Zmapqtl outputs a great deal of information: Often the experimenter will want a quick summary of the positions and effects of the QTLs. The program **Eqtl** scans the output of **Zmapqtl** and reformats it. Part of the output of **Eqtl** is identical to the output format of **Rqtl**. This is convenient if the experimenter would like to do simulation studies with a set of estimated QTLs. The output of **Eqtl** can be used as the input to **Rcross** (with the appropriate genetic linkage map), and new data sets can be simulated to examine the power of the different methods to detect the QTLs. Finally, the output of **Eqtl** can be read by **Zmapqtl** and used to create virtual markers to be used as covariates in composite interval mapping (see model seven of Section 3.4.2).
The remaining output is more readable and is appropriate if the experimenter is not interested in doing further simulations. The positions of the QTL are given in Morgans from the telomere rather than recombination frequencies from the flanking markers.
In addition to reformatting the output of **Zmapqtl**, **Eqtl** will automatically detect whether a permutation test, jackknife or bootstrap experiment had been done. If such results exist, **Eqtl** will open and summarize them. For example, if you do a permutation test with

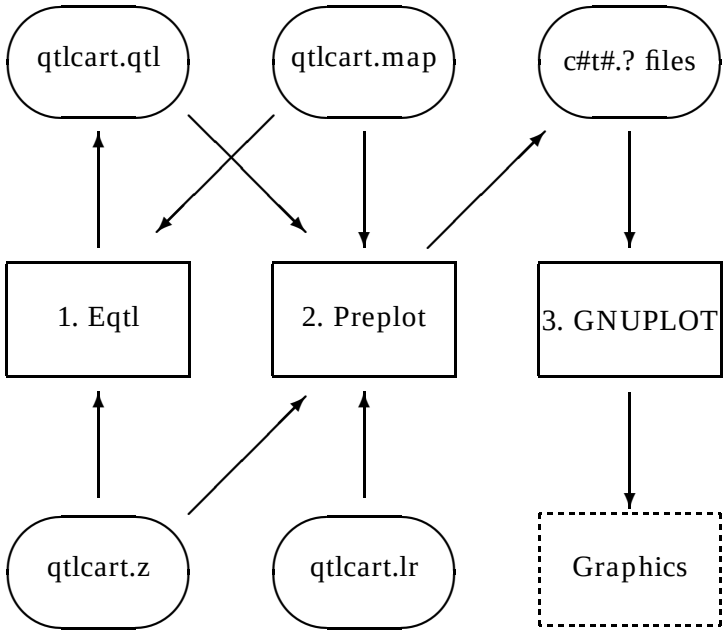

Figure 4.1: Visualization Schematic

Zmapqtl using interval mapping, an interim file **qtlcart.z3e** is created and appended to for each permutation. **Eqtl** will read this file and calculate experimentwise threshold values from it. Standard significance thresholds will be written to the log file. The user can specify a type I error rate (size) and **Eqtl** will calculate a threshold value relevant to it. Once done, the threshold value will be remembered and used by subsequent runs of **Eqtl** or **Preplot**.

For bootstrap results from **Zmapqtl** using interval mapping, **Eqtl** looks for a file **qtlcart.z3a**. If found, **Eqtl** will read in the sums and sums of squares of the likelihood ratio, additive effect and dominance effect at each position and print the mean and sample standard deviations into a summary file (**qtlcart.z3d**). **Eqtl** does similar calculations for the jackknife results that would be in **qtlcart.z3i**.

Table 4.1 shows the command line options specific to Eqtl.

4.1.1 Options

Files

Similar to other programs in the ***QTL Cartographer*** system, the input and output files can be specified. A genetic linkage map and a file containing the results of **Zmapqtl** must exist and be properly specified to **Eqtl**.

Option	Default	Explanation
-z	qtlcart.z	(Composite) Interval Mapping Results
-o	qtlcart.eqt	Output File
-m	qtlcart.map	Genetic Linkage Map File
-M	3	Model from Zmapqtl
-H	1	Hypothesis Test (30,31,32) for F2
-S	10.0	Significance threshold
-a	0.05	Size (α)
-L	0	Output LOD scores? (0=no,1=yes)

Table 4.1: Command Line Options for Eqtl

Which Results?

The output file from **Zmapqtl** may contain the results of analyzing different traits using different models. Furthermore, in F_2 and other populations in which dominance can be estimated, it is possible to test different sets of hypotheses. The user can specify which results from the **Zmapqtl** output file to process. The *-M* option tells **Eqtl** to examine the results from using the specified analysis model. An integer value should be given after the *-M* option. By default, **Eqtl** looks for the results from Model 3, or interval mapping. If you have done composite interval mapping, with say model 6, then you should specify *-M 6* on the command line (or in the interactive menu). If model 6 was the last model run in **Zmapqtl**, then **Eqtl** should already be aware of that fact.
The output file may also contain results from different traits. The default trait is 1, but can be changed with the *-t* option. Of course, some users may choose to have a different output file for each trait in turn, and then the *-z* and *-t* options should be used together. Remember that at the beginning of each set of results in the **Zmapqtl** output file, the trait is specified. **Eqtl** looks to match this.
For F_2 design experiments, various hypothesis tests can be performed. These are explained in the previous chapter. Using *-H* with an integer allows you to specify which hypothesis test results to use. Presently, the choices are 1, 2 and 3 for the $H_3 : H_0$, $H_3 : H_1$ and $H_3 : H_2$ comparisons, respectively.

Other Options

Eqtl essentially finds the peaks in the graph of the results from **Zmapqtl**. It goes along the chromosome, and determines whether the likelihood ratio test statistic is increasing or decreasing. Upon a change, it picks out the position and estimates of other parameters. The user can specify that the peaks of interest need be higher than some "Significance threshold" to be considered QTLs. The default is 3.84, that is, any peak that is less than 3.84 is ignored. This can be changed with the *-S* option. If you have run **Zmapqtl** and done a permutation test, **Eqtl** automatically reads the output and sets the significance threshold subject to the value of the size, set with the *-a* option. For a size of α, the the $100(1-\alpha)$-

percentile is calculated from the experimentwise test values.
The final option is a flag to output LOD scores rather than likelihood ratios. The default behavior of the ***QTL Cartographer*** system is to use a likelihood ratio test statistic (LR) rather than a LOD score. For a hypotheses H_i, let L_i be the likelihood of the data given the hypothesis. For a pair of hypotheses H_0 and H_1, this would yield L_0 and L_1. The LOD score is defined as

$$LOD = -\log \frac{L_0}{L_1}$$

The likelihood ratio test statistic (LR) is

$$LR = -2\ln \frac{L_0}{L_1} = -2\ln 10^{-LOD} = 2(\ln 10)LOD = 4.605LOD$$

and thus

$$LOD = -\log \exp(-\frac{LR}{2}) = \frac{1}{2}(\log e)LR = 0.217LR$$

4.2 Preplot

Preplot reformats the output of the analysis programs so that they may be plotted by **GNUPLOT**. The output files could be imported into any programs. The default behavior of **Preplot** is what we term the "automagic" mode. **Preplot** reads the **Zmapqtl** output file, determines what analyses have been done, and then reformats all of these analyses in a logical way. There will be a separate graph for each trait and each chromosome. **Preplot** will attempt to put the results from different models in **Zmapqtl** and from **LRmapqtl** on the graphs, along with any information from the **Rqtl** output file (if it exists), and a significance threshold (which can be set in the interactive menu or on the command line). Table 4.2 shows the command line options specific to **Preplot**. In general, it will not be necessary to change any options to **Preplot**. Most of the proper values should have been set by other programs in the ***QTL Cartographer*** suite. You might want to use the *-L* command to tell **Preplot** to convert LR values into LOD scores. In any case, the output of **Preplot** is ready for import into **GNUPLOT**. There will be a number of output files. One is a plot control file, that has commands that **GNUPLOT** understands. The other files simply contain two columns of numbers for the x and y coordinates to plot. The names of the files indicate what the numbers are for. They all start with a lower case *c* , which indicates chromosome. Following the *c* is an integer indicating which chromosome, then there is a *t* followed by an integer indicating the trait. Then there is a period and a file extension that indicates the results contained in the file. For the results of composite interval mapping, the ".z" filename extension will be followed by an integer from 1 to 7 indicating the model used for the analysis. For example, the file **c2t3.z6** would have the results of composite interval mapping for trait 3 on chromosome 2 in it.

4.2.1 Printing Results

One option that is useful to change is the "Terminal" setting. This will be set correctly if all you want to do is view the graphs on your screen with **GNUPLOT**. If you want to

Option	Default	Explanation
-o	qtlcart	Gnuplot Control File Name
-m	qtlcart.map	Genetic Linkage Map File
-q	qtlcart.qtl	QTL or Estimated QTL file
-l	qtlcart.lr	LRmapqtl Output File
-z	qtlcart.z	Zmapqtl Output File
-S	10.0	Significance Threshold
-T	x11	Terminal
-L	0	Output LOD scores? (0=no,1=yes)
-i	1	Hypothesis (for F2 design)

Table 4.2: Command Line Options for Preplot

get a hardcopy printout, you have two alternate choices for the "Terminal" option. If you have a postscript printer, then use "postscript" as the terminal. Run **Preplot**, and then run **GNUPLOT** as explained in Section 4.3. You will not see any output, but a file **qtlcart.ps** will be created (or **stem.ps**, where "stem" is your filename stem). This file can be sent to any postscript printer. The other alternative is "hpljii", which does something similar for HP-LaserJet II's (the output file will be **stem.hp**). You could use the "hpljii" option for "Terminal", and then edit the **stem.plt** file to change the type of printer to anything that **GNUPLOT** supports: See the **GNUPLOT** manual (Williams and Kelley 1993) for more details.

Extension	Meaning
s	Significance Threshold
lr	Linear Regression results
z#	Composite interval mapping results
q	Quantitative trait locus data (from Rqtl)

Table 4.3: Filename extensions for Preplot output

4.3 GNUPLOT

GNUPLOT is free plotting software available for UNIX, Macintosh and Windows machines. It is an interactive package. The basic idea behind the program is to read in simple files of numbers and plot them. The files of numbers contain two columns, one for the abscissa and one for the ordinate. Preplot takes care of reformatting the output of the analysis so that **GNUPLOT** can read the results and plot them. We have placed copies of **GNUPLOT** for the three platforms on our ftp server.

4.3.1 Basic GNUPLOT

In many ways, **GNUPLOT** is similar to **MAPMAKER** in that it is an interactive, command driven program. Once **GNUPLOT** has been started, the user can type "help" to get information on how to use the program. There are commands to change the terminal type, load files and specify the output device. Thus, one can view or print the images created by **GNUPLOT**.
If you have run **GNUPLOT**, you should have a plot control file with a ".plt" extension. Suppose that this file was **stem.plt**. You can start up **GNUPLOT** and issue the command

```
gnuplot> load "stem.plt"
```

to see the plot specified by **stem.plt**.
See the **GNUPLOT** manual for more information on this program (Williams and Kelley 1993). Of special interest may be the different types of printers supported by **GNUPLOT**. If you choose "postscript" as your terminal type in **Preplot**, then you will find a pair of lines on the **stem.plt** file that look like this:

```
set term postscript
set output "stem.ps"
```

You can change the token "postscript" token in that file to any printer that **GNUPLOT** supports and sent the **stem.ps** file to that printer.

Chapter 5

Tutorial Examples

5.1 General tactics and notes

Below, we outline some general exercises using ***QTL Cartographer*** . These exercises were used in a class (Statistics 591o) and in the Summer Institute in Statistical Genetics at North Carolina State University. These computer exercises were done in the Statistics Instructional Computing Laboratory (SICL), which is equipped with Sun workstations running Solaris 2.5, but the exercises can be done on any platform that ***QTL Cartographer*** runs on. As a general rule, we suggest creating a separate subdirectory (folder) for each data set. Copy the original input files into that subdirectory. This will help to organize your work. In addition, since you will be working with copies, your original files will be safe.
Beginning with the first program you run, a resource file called **qtlcart.rc** is created and updated for each subsequent program. This file keeps track of all the parameters and file names that you use. In addition, a log file will record which specific parameters were used with which specific programs, and when the programs were run. Thus, the **qtlcart.rc** file keeps track of the current settings, and the **qtlcart.log** file records the history of parameter settings. You can look at any of these files or any other files that ***QTL Cartographer*** creates by opening them in any text editor.
Macintoshes and PCs work a little differently than the examples below. They maintain one copy of the **qtlcart.rc** file in the subdirectory (folder) where the applications are located. You can specify a working subdirectory (folder) in any of the ***QTL Cartographer*** programs, and this will be recorded in the **qtlcart.rc** file. The Introduction has more extensive instructions on how to do this. If you are on a Macintosh or a PC, create a subdirectory (folder) called "qwork" in the subdirectory (folder) where the binaries are. If you are on a UNIX machine, create the "qwork" subdirectory in your root directory.
There is a web page for ***QTL Cartographer***

`http://statgen.ncsu.edu/qtlcart/cartographer.html`

which is the good place to keep abreast of new information. The "readme" file from the ftp server is linked to the web page. The programs are also linked to the web page, so you can download them using some web browsers. The entire manual as well as the man pages have been been translated into html.

5.2 Basic Macintosh

The MacOS is so easy to use that little instruction is necessary. I would recommend getting a copy of **BBEdit Lite** for viewing and editing text files. It is freeware and can open large files as long as you have the memory. **BBEdit Lite** can also view and convert text files with DOS, UNIX or Macintosh line endings. Other free programs such as **Fetch** to download files, **Telnet-2.7** to access UNIX servers and **Acrobat Reader** to view and print documents are also useful.

5.3 Basic Windows

This is a quick summary of some basic commands and techniques for working in the Windows NT environment: Other versions of Windows from 95 up should be similar.

Logging in Using "control+alt+delete" will bring up the login screen. Click in the login box and type your login name. Press tab to get to the password box and type your password and a return. If you then see a timer (which looks like a little clock), you'll know you have succeeded: Just wait while the windowing system starts up.

Logging out You may want to empty the trash before you log out. Right-click on the recycle bin and select "empty" to do so. When you want to log out, simply click the left mouse button on the **Start** icon and select shutdown. This will bring up a menu. Select the "Close all programs and logon as a different user" option and then click "OK".

5.3.1 Navigating disks

I generally use **Windows Explorer** to navigate the disks. You can click on files, copy them and paste them in different directories to make copies of files. If you are not familiar with **Windows Explorer**, take a few minutes to play with it. You can double click on the "My Computer" icon and icons therein to explore your hard drive.

Viewing files

There are a lot of options for viewing files. Generally I recommend using **Notepad**. It is a simple text editor with a fixed width font. You can find it under "Start → Programs→ Accessories → Notepad". When you try to open a file, be sure to tell **Notepad** to look for files of type "All Files". If you don't, then **Notepad** will only show files with a ".txt" extension.
Windows NT does not like files to be accessed by two programs at once. Be sure to clear out **Notepad** by creating a new file before running any programs that might read or write to a file that you are viewing.

Command Prompt

Clicking on "Start → Programs → Command Prompt" brings up a command line window for DOS commands. You can **ftp** or **telnet** from this window if you wish to transfer files or logon to an account elsewhere. There is a text editor that can be started with the command **edit** that will allow you to view files. Again, take care not to open files that are being accessed by other programs.

Transferring Files

You can start a **Command Prompt** and from there ftp files to your home account. You will need the IP number or hostname and domain name to do this. Simply start up the **Command Prompt**, type in the drive from which you want to transfer files, and **cd** to the directory where the files are. Then, **ftp** to your home machine and put the files there. Use "quit" to kill ftp and "exit" to get back to Windows. Here is an example.

```
c:\>  k:
k:\>  cd module5
k:\module5> ftp mymachine.somedomain.net
ftp> prompt
ftp> mput *
ftp> quit
k:\module5> exit
```

5.4 Basic Unix

This is meant to be a quick summary of some basic Unix commands. One thing to keep in mind is that Unix is case sensitive. Feel free to practice any of the following commands (but be careful with **rm** and **mv**).

5.4.1 Help!

The **man** command is one of the most important for the novice and experienced user. If you would like to know what it does, type **man man** at the prompt in a command window. You can use it to get information on most of the commands below.

5.4.2 Basic filesystem commands

Here is a list of basic commands for seeing, copying and moving the files in your directory, creating new subdirectories and navigating. Go ahead and experiment with these commands.

- **ls** is a command to list the files in the present working directory. You can give it options, for example **ls -l** will give listings with more information about the files than **ls**.

- **pwd** tells you where you are. This can be useful if you have created many subdirectories.
- **cd** allows you to change the current working directory. You can give it an absolute or a relative argument. **cd ..** would move you to the next highest subdirectory. **cd /ncsu/pams046/bin** would move you to the the /ncsu/pams046/bin subdirectory, *etc.*
- **mkdir** allows you to create a subdirectory. **mkdir test** would create the subdirectory *test*. **rmdir** would remove it. You can only remove empty subdirectories.
- **rm** allows you to remove a file. It is aliased as **rm -i**, which means that it will ask if you really want to remove the file. **rm filename** would remove the file *filename*.
- **mv** moves a file. **mv file.orig file.new** would move the file *file.orig* to *file.new*. You can think of it as renaming.
- **cp** copies one or more files. **cp file1 file2** copies the file *file1* to the file *file2*. **cp file1 file2 direct** would copy the files *file1 file2* to the directory *direct*.
- **chmod** is a rather complex command to change the permissions on files. You can write batch files, and use **chmod** to allow execution of them.
- **more** will display the contents of a file. Use it as **more filename**. While in **more**, typing a **q** will get you out.

5.4.3 Curious?

There are a couple of commands to find out who is on your machine and what they are doing. **w**, **who** and **finger** tell you who is logged on to your machine and what they are doing. You can also use **finger** to get info on a user on another machine. Try **finger basten@statgen.ncsu.edu**.

5.4.4 Other commands

- **rlogin**, **telnet** and **ftp** allow you to initiate sessions on other machines. You need to supply the IP address or nickname of the machine with these commands..
- **exit** closes a terminal window and **clear** clears it.
- **history** shows the last 40 commands issued. They will be numbered, and you can rerun them with an exclamation point and the number of the command, *e.g.*, **!23** would run the command numbered 23 in the **history** list.
- **lpr** sends a file to the printer. You can print up to 50 sheets from your account.
- **alias** allows you to assign Unix commands to more familar words. For example, **alias dir ls** would allow you to type **dir** to list the files in a directory. **alias** with no arguments would list the current aliased commands.

5.5 Simulating and Analyzing data

Assuming that you have a "qwork" subdirectory (folder), create a new subdirectory (folder) within it. You can call it anything you like, but for the purposes of illustration it will be referred to as "example1". Thus, your working directory for this example will be "qwork:example1" if on a Macintosh or "qwork\example1" if on a PC. If you are on a UNIX machine, cd into the "qwork/example1" subdirectory and don't worry about setting a working subdirectory. Also note that if you are using a PC, the program names will all have an ".exe" ending.
In this exercise, you will simulate a genetic linkage map, then a model and finally a data set. This data will then be analyzed.

1. Start up **Rmap**. Select the option to change the filename stem. Change the filename stem to "sim". You can change any parameters that you like. We suggest changing the variances of markers per chromosome and intermarker distance to values other than 0.0. In each case, a value of 2 or 3 would work well for the purposes of this exercise. Don't change the output format. If you are on a Macintosh or MS-Windows machine, be sure to set the proper working subdirectory (folder). When satisfied with the parameter values, select "0" to run the program. Look at the output (sim.map).

2. Start up **Rqtl**. You probably don't need to change any parameters. You can run this program with the "0" option. Look at the output (sim.qtl).

3. Start up **Rcross**. Again, you do not need to change any parameters, but you could try a different experimental design. Select the number associated with the experimental design. Change its value from "B1" to "SF3" (or whatever you like from Table 1.1). Run this program with the "0" option. Look at the output (sim.cro). From this point on, the analyses will utilize this file and the "sim.map" file.

4. Start up **Qstats** and run it without changing any parameters. Look at the output (sim.qst).

5. Start up **LRmapqtl** and run it without changing any parameters. Look at the output (sim.lr).

6. Start up **SRmapqtl**. You might want to change the analysis model from its default value of "0" (forward stepwise regression) to "2" (forward regression with backward elimination). Run it and look at the output (sim.sr).

7. Start up **Zmapqtl**. You won't need to change any parameters. Tell it to go ahead with the analysis and look at the output (sim.z).

8. Start up **Zmapqtl** again. This time, choose "Model for Analysis" and change it to "6". Tell it to go ahead with the analysis, and look at the output (which will be appended to what you did in the first run).

9. Start up **Preplot**. Don't change any parameters: Go ahead with the program.

10. If you are on a Macintosh, move the **GNUPLOT** binary into the working subdirectory and double click it. If on PC or UNIX machine, start up **GNUPLOT**. From the **GNUPLOT** command line, type in
    load "sim.plt"
    (If you are on a PC, you may need to go through the file menu and search for the "sim.plt" file.) This should display graphical results. Press returns when requested.

11. Start up **Eqtl**. Go ahead with the analysis. Look at the output (sim.eqt).

5.6 Analyzing simulated data

Create a working subdirectory (call it "mletest") and copy the simulated data sets into it. The simulated datasets called "mletest.map" and "mletest.cro" come from Zeng (1994). They are in the "doc" folder for the Macintosh versions, in the same place as the binaries on PCs and in the "example" subdirectory in the UNIX version. These are properly formatted, and can be analyzed with **Qstats**, **LRmapqtl**, *etc.* Do the following:

1. Proceed with the analysis programs as in the previous example. Be sure to set the proper filename stem (mletest) and working subdirectory. Run **Qstats**, **LRmapqtl**, **SRmapqtl** and **Zmapqtl**. Look at the output after each run.

2. Start up **Preplot**. Don't change any parameters: Go ahead with the program.

3. Start up **GNUPLOT**. From the **GNUPLOT** command line, type in
    load "mletest.plt"
    This should display graphical results. See the first example for the specifics of PCs and Macintoshes.

4. Start up **Eqtl**. Go ahead with the analysis. Look at the output (mletest.eqt).

5.7 Analyzing real data

Create a new working subdirectory called "realdat" in you "qwork" subdirectory. Copy the realdat*.inp files into it. There should be two files: "realdatm.inp" and "realdatc.inp". The former is a genetic linkage map in the standard input format (map.inp). The latter is a file with marker and trait data in the standard input format (cross.inp). This is a real data set kindly provided by Juan Medrano (Horvat and Medrano 1995). It has also been used as an example in a review on the statistical issues in QTL mapping (Doerge, Zeng, and Weir 1997). You will now translate the data files into the ***QTL Cartographer*** format, and then analyze the data.

1. Start up **Rmap**. Change the working subdirectory, and then the filename stem. You can use "realdat" for the stem. Now, select item 1 from the menu and enter "realdatm.inp". Now run the program. **Rmap** should read in the prepared genetic linkage map file and reformat it properly.

2. Start up **Rcross**. Select item 1 from the menu and enter "realdatc.inp". Now run the program. **Rcross** should read in the prepared data file, match marker names from this data file to those in the map file, and reformat the data properly. Look at the output.

3. Proceed with the analysis programs as in the previous examples. Run **Qstats**, **LRmapqtl**, **SRmapqtl** and **Zmapqtl**. Look at the output after each run.

4. Start up **Preplot**. Don't change any parameters: Go ahead with the program.

5. Start up **GNUPLOT**. From the **GNUPLOT** command line, type in
   load "realdat.plt"
   This should display graphical results.

6. Start up **Eqtl**. Go ahead with the analysis. Look at the output (realdat.eqt).

5.8 Analyzing a MAPMAKER data set

5.8.1 Using MAPMAKER/EXP

You will need **MAPMAKER/EXP** for this part. If you don't want to use **MAPMAKER/EXP**, then you can use the already prepared files that come with the distribution. Otherwise, ftp to genome.wi.mit.edu and **cd** to **/distribution/mapmaker** to get the programs. A file **sample.raw** comes with **MAPMAKER/EXP**.
Each number is a command in a sequence to be done in **MAPMAKER/EXP**. Anything inside of square braces are comments and should not be typed into **MAPMAKER/EXP**. Start up **MAPMAKER/EXP** in an appropriate subdirectory and proceed with these commands:

1. prepare data sample.raw [Input the data from the raw file.]
2. photo sample.tutorial [Save what you do in a log file.]
3. sequence 1 2 3 4 5 6 7 8 9 10 11 12 [Start with all markers.]
4. group [Group them into linkage groups]
5. sequence { 1 2 3 5 7 } [Use randomly ordered group 1 makers.]
6. compare [Compare all orders. For each in turn, calculate the Likelihood.]
7. sequence 1 3 2 5 7 [Decide that this is the best order and specify it.]
8. map [Print the map to the screen. This attaches distances as well.]
9. sequence 4 6 8 9 10 11 12 [Now use the rest of the markers.]
10. list loci [Summarize the number of informative progeny.]
11. lod table [Show pairwise distances and linkage LOD scores.]

12. sequence {8 9 10 11 12} [Use a randomly ordered subset of markers from group 2.]
13. compare [Compare all orders. For each in turn, calculate the Likelihood.]
14. sequence order1 [Use the best order from the compare command.]
15. try 4 6 [Try all possible positions of markers 4 and 6. Also, try unlinked idea.]
16. sequence 4 11 8 12 9 6 10 [This is the best sequence.]
17. make chromosome c1 [Create chromosome 1.]
18. sequence 1 3 2 5 7 [Specify the sequence of markers on chromosome 1.]
19. attach c1 [Attach the sequence to chromosome 1.]
20. framework c1 [Create the framework (puts in distances) for chromosome 2.]
21. make chromosome c2 [Create chromosome 2.]
22. sequence 4 11 8 12 9 6 10 [Specify the sequence of markers on chromosome 2.]
23. attach c2 [Attach the sequence to chromosome 2.]
24. framework c2 [Create the framework (puts in distances) for chromosome 2.]
25. quit [Exit the program. The map will be in sample.maps.]

On a UNIX machine, you will now have a file called **sample.maps**. On a PC, it will be called **sample.map**. It will be one of these two on a Macintosh. Rename this output file to **sample.mps**, and use it along with the **sample.raw** file for the next part.

5.8.2 Using the MAPMAKER files

Create a new working subdirectory called **mm** in you **qwork** subdirectory. Copy the sample.* files into it. There should be two files: **sample.mps** and **sample.raw**. The former is a genetic linkage map created by **MAPMAKER/EXP**. The latter is **MAPMAKER/QTL** raw file. You will now translate the data files into the ***QTL Cartographer*** format, and then analyze the data.

1. Start up **Rmap**. Select the option to change the filename stem. Change the filename stem to “sample” and set the proper working subdirectory. Then select the input file option and change it to “sample.mps”. Then go ahead with the analysis. Look at the output (sample.map).
2. Start up **Rcross**. Select the input file option and change it to “sample.raw”. Then go ahead with the analysis. Look at the output (sample.cro).
3. Proceed with the analysis programs as in the previous examples. Run **Qstats**, **LRmapqtl**, **SRmapqtl** and **Zmapqtl**. Look at the output after each run.

4. Start up **Preplot**. Don't change any parameters: Go ahead with the program.
5. Start up **GNUPLOT**. From the **GNUPLOT** command line, type in
   load "sample.plt"
   This should display graphical results.
6. Start up **Eqtl**. Go ahead with the analysis. Look at the output (sample.eqt).

Chapter 6

Input File Formats

All of the input and output files in the ***QTL Cartographer*** system are plain text, and can thus be viewed by virtually any text editor or word processor on any platform. The input files for many of the programs will have embedded commands that start with a minus sign (-). Care should be taken not to have stray tokens such as '-Chromosome' in input files. Also, the case of commands is generally very important: When in doubt use the exact case that is specified here.

6.1 Genetic Linkage Maps

6.1.1 MAPMAKER output files

Rmap can translate the output of **MAPMAKER** into the format required by the ***QTL Cartographer*** system. Use the .maps file that is the output of **MAPMAKER** as the input to **Rmap** and it will be translated automatically. An alternate format has been designed for those who don't have the **MAPMAKER** files.

6.1.2 Rmap input files

The general method of inputting data for this format is by tokens. Tokens are just collections of characters surrounded by whitespace (spaces, carriage returns, tabs, line feeds). The maximum length of any token must be less than 64 (and this may be increased in the future).
The following file also has commands embedded into it. **Rmap** recognizes any token that begins with a minus sign (-) as an embedded command. Some commands require that the following token be a number or piece of information. The following table gives a list of tokens that the program recognizes, their purpose and what the next token should be.

- -type Defines what the distances will be. The token following this command must be either *positions* or *intervals*. The latter indicates that the numbers are for the interval distance after a marker, while the former indicates a position from the left telomere.

- -function Defines a mapping function. It can take on the values 1, 2 or 3 for the Haldane, Kosambi and Complete interference functions, respectively
- -Units Indicates the units of the distances. Valid tokens following this command are *cM, M* or *r* for centiMorgans, Morgans or recombination probabilities. Case is important.
- -chromosomes Indicates the number of chromosomes. The following token must be an integer equal to the number of chromosomes in the map.
- -maximum Should be followed by an integer indicating the maximum number of markers on any chromosome.
- -end or -quit indicate to **Rmap** that it should stop reading from the file.
- -named Should be followed by *yes* or *no* indicating whether the marker systems have names.
- -start indicates the start of the genetic map.
- -stop indicates the end of the genetic map.
- -skip tells **Rmap** to ignore all tokens until an '-unskip' token is encountered.
- -unskip see above.
- -Chromosome should be followed by an integer indicating the chromosome number.

The first line should start with a # and have some long integer after it. After that it should have the token 'bychromosome'. The number will be an identifier for the file, and should be unique. The token 'bychromosome' indicates how the map should be read in. Here is an example of a first line:

```
# 123456789 bychromosome -filetype map.inp
```

The final pair of tokens indicate what type of file it is.
Between the '-start' token and the '-stop' token, you should have a repeating sequence of a '-Chromosome' token, an integer, then markers ordered, with their names followed by the appropriate distances. This example has the markers followed by their positions in centiMorgans. All markers should have unique names.

```
-start
-Chromosome 1
   Marker1_1    0.0
   Marker1_2   10.2
   Marker1_3   34.1
   Marker1_4   43.3
   Marker1_5   52.1
-Chromosome 2
   Marker2_1    0.0
```

```
   Marker2_2   13.7
   Marker2_3   19.1
   Marker2_4   24.8
-stop
```

You can annotate the input file as much as you want. Just don't put in any extra material before the '-stop' token. Everything after the '-end' token is ignored. Before the '-start' token, only the -type, -function, -units, -chromosomes and -maximum tokens are processed. The token following each of these is read and the information used in the program.
The format of the information between the -start and -stop commands is unimportant. You just need whitespaces around each piece of information. All the marker names and their distances could be on one line.
If this file were called **map.inp**, then
`% Rmap -A -V -i map.inp`
would convert this file to the format required for the other programs in the ***QTL Cartographer*** system.
If '-named' had a value of 'no' above, then the format of the distances would be:

```
-start
-Chromosome 1  0.0 10.2 34.1 43.3 52.1
-Chromosome 2  0.0 13.7 19.1 24.8
-stop
```

6.1.3 Rmap output files

Rmap overwrites any file that has the same name as specified as the output file. Be careful not to destroy any important files. The output file will contain the values of the parameters used, the names of chromosomes and markers (if a translation was made) and the linkage map.

6.2 QTL information

You can specify a genetic model and use it for simulation by translating it with **Rqtl**. This would be useful if you want to do some "what-if" experiments.

6.2.1 Rqtl input files

The input format is similar to that for **Rmap**. The input is token based. The first line should start with a pound symbol and have a long integer after it. The number will be an identifier for the file, and should be unique. Finally, a filetype should be speciefied on the first line.
`# 12345789 -filetype qtls.inp`
These commands are recognized

- -Units Indicates the units of the distances. Valid tokens following this command are *cM*, *M* or *r* for centiMorgans, Morgans or recombination probabilities.

- -named indicates whether traits will have names. Valid tokens following this command are *yes* and *no*.
- -skip Begin skipping tokens until an -unskip token is encountered.
- -unskip see above
- -start start data segment
- -stop stop reading data
- -end, -quit close file.

After the start token, there should be the token 'qtls' and a number to indicate the number of traits to be modeled. After this there should be a repeating sequence of a trait name, number of loci for that trait, then the chromosome, position, additive and dominance effects for each locus. This example has the loci followed by their positions in centiMorgans (from the telomere). Please give all traits unique names.

```
-start qtls 3
Trait_1 4
    1   9.1  0.75  0.0
    1  89.1  0.5   0.0
    3  68.4  0.22  0.0
    4  43.2  0.95  0.0
Trait_2 2
    2  93.4  0.42  0.0
    4  33.2  0.90  0.0
Trait_3 1
    1  33.4  0.84  0.2
-stop qtls
```

The format of the information between the -start and -stop commands is unimportant. You just need whitespaces around each piece of information. All the marker names and their distances could be on one line. Note that in the above example for Trait_1, there are two QTL on chromosome 1.
If this file were called 'qtls.inp', then
`% Rqtl -A -V -i qtls.inp`
would convert this file to the format required for the other programs in the ***QTL Cartographer*** system.

6.2.2 Rqtl output files

Rqtl overwrites any file that has the same name as specified as the output file. Be careful not to destroy any important files. The output file will contain the genetic model in a format suitable for input into **Rcross**.

6.3 Data files

These are files that contain marker and trait data. The output format of **Rcross** is rather difficult for the user to read and create manually. We have therefore provided ways to translate other formats.

6.3.1 MAPMAKER raw files

Rcross will convert **MAPMAKER** raw files for use in the ***QTL Cartographer*** system. You will first need to use **MAPMAKER** to create a genetic linkage map. Then convert the map into the "Rmap.out" format for use with**Rcross**. Then, use **Rcross** to convert the **MAPMAKER** raw data file into the "Rcross.out" format.

6.3.2 Rcross input files

We have also defined a format for your data. It is similar to the input formats for **Rmap** and **Rqtl**. Input is token based, and the data file has embedded commands to indicate to **Rcross** what it is reading. The first line of the data file should contain a pound symbol and a long integer, for example

```
# 123456787 -filetype cross.inp
```

The number will be an identifier for the file, and should be unique to this file. In addition, the '-filetype cross.inp' token helps **Rcross** determine what type of file it is reading.
Here is a list of embedded commands:

- -skip indicates that **Rcross** should skip all tokens until an -unskip token is read.
- -unskip see above
- -Cross Should be followed by the type of cross. See Table 1.1 for valid tokens.
- -traits should be followed by the number of traits that have numerical values.
- -otraits should be followed by the number of other traits, that is those with character or string values. Examples would include sex or brood.
- -SampleSize would be followed by the sample size.
- -case should be followed by *yes* or *no* depending on whether the names of marker systems are case sensitive. With *no,* all names of individuals, markers and traits are converted to lower case to make comparisons.
- -TranslationTable will allow one to define a table to translate marker values. After this command, a small table of six rows and three columns must follow. The first two columns should match exactly the example given below, and the third column can be whatever your data set is encoded as.
- -missingtrait followed by a token indicates that when reading trait data, the given token indicates missing phenotypic data.

Translation Table

This is an example of a translation table for marker information.

```
-TranslationTable
     AA    2    2
     Aa    1    1
     aa    0    0
     A-   12   12
     a-   10   10
     --   -1   -1
```

Note a few things in the above translation table. There are six rows and three columns. There must be a token in all 18 positions of the table. The first column is the genotype. The program assumes that the 'A' allele is diagnostic for the 'High' (Parental 1) line and the 'a' allele is diagnostic for the 'Low' (parental 2) line. (These were previously denoted by A_1 and A_2: They aren't here because the above text comes from an ascii file.) A minus sign '-' means the allele is unknown. Thus, dominant markers can be encoded. The middle column is how the output of these genotypes will be encoded while the right (3rd) column is how you will code the input of this file. The above TranslationTable maps 2 to 2, 1 to 1, 0 to 0, etc. Just about any set of tokens can be used for the third column, but DO NOT change the first two columns. If you encoded your P1 homozygotes as BB, heterozygotes as Bb, etc, your translation table might appear as

```
-TranslationTable
     AA    2   BB
     Aa    1   Bb
     aa    0   bb
     A-   12   B-
     a-   10   b-
     --   -1   --
```

Anything in the following data file that is not recognized (doesn't match something in column (3) will become unknown (-1) in the output.

Data by Markers and Traits

One way to organize the data is by markers. For each marker, you give the genotypes of the individuals. The order of the individuals has to be the same for each marker. Below is an example. After the '-start markers', the program expects a repeating sequence of marker name, then n marker genotypes where n is the sample size. The marker names should match those in the **map.inp** file.

```
-start markers
Marker1_1 2 2 2 2 2 1 1 1 1 1
Marker1-2 2 2 2 1 1 1 1 1 1 1
```

```
Marker1_3 1 2 2 2 2 1 1 1 1 1
Marker1_4 1 1 2 2 2 1 1 1 1 1
Marker1_5 2 2 2 2 2 1 1 1 1 1
Marker2_1 2 1 1 2 2 1 1 1 1 1
Marker2_2 2 2 2 1 1 1 1 1 1 1
Marker2_3 2 2 1 1 1 1 2 2 2 2
Marker2_4 2 1 1 1 1 1 1 1 2 2
-stop markers
```

The traits are encoded in the same fashion. After the '-start traits' tokens, the program expects a repeating sequence of trait name and then n values for the sample. The order of the individuals has to be the same as in the markers. In the following example, a period indicates missing trait data.

```
-missingtrait .
-start traits
Trait_1    5.0  5.3  6.2  4.1  5.5  5.8  6.7  6.1   .   6.4
Trait_2   15.0 15.3 16.2 24.1 25.5 25.8 16.7 26.1 33.2 16.4
-stop traits ... indicates the end of the trait data.
```

Other traits (otraits) will be stored as character strings. These will be things such as sex, brood, eye color, *etc.* Each token should be less than 64 characters in length. They should have a repeating order that is the same as the trait data, and the "-missingtrait" command is recognized.

```
-missingtrait .
-start otraits
Sex   M F M M M F F M F F
Brood 1 1 1 1 1 2 2 2 2 2
-stop otraits
```

Data by individuals

Another way to organize the data is by individuals. The program expects that the markers are ordered from marker 1 on chromosome 1, marker 2 on chromosome 1, ..., to the last marker on the last chromosome. Since the individuals are named, they can be in any order.

```
-start individuals markers
Ind_1  2 2 1 1 2 2 2 2 2
Ind_2  2 2 2 1 2 1 2 2 1
Ind_3  2 2 2 2 2 1 2 1 1
Ind_4  2 1 2 2 2 2 1 1 1
Ind_5  2 1 2 2 2 2 1 1 1
Ind_6  1 1 1 1 1 1 1 1 1
Ind_7  1 1 1 1 1 1 1 2 1
Ind_8  1 1 1 1 1 1 1 2 1
```

```
Ind_9  1 1 1 1 1 1 1 2 2
Ind_10 1 1 1 1 1 1 1 2 2
-stop individuals markers
```

The traits are done similarly. All the traits have to be in these types of blocks, but you can have more than one block. Each column is for a different trait. After the '-start' token, put individuals followed by traits, then the number of traits (2, in this case), then the names of the traits, then indicate whether the individuals are named. Here they are named, but if they weren't, put an 'notnamed' token where the 'named' token presently is. Other traits follow a similar pattern, and an example is given below.

```
-start individuals traits 2 Trait_1 Trait_2 named
Ind_1  5.0 15.0
Ind_2  5.3 15.3
Ind_3  6.2 16.2
Ind_4  4.1 24.1
Ind_5  5.5 25.5
Ind_6  5.8 25.8
Ind_7  6.7 16.7
Ind_8  6.1 26.1
Ind_9   .  33.2
Ind_10 6.4 16.4
-stop individuals traits

-start individuals otraits 2 sex brood named
Ind_1   M 1
Ind_2   F 1
Ind_3   M 1
Ind_4   M 1
Ind_5   M 1
Ind_6   F 2
Ind_7   F 2
Ind_8   M 2
Ind_9   F 2
Ind_10  F 2
-stop  individuals otraits
```

Chapter 7

Benchmarks

Tables 7.1-7.2 summarize the timings for **Zmapqtl** to do interval mapping (Lander and Botstein 1989) and composite interval mapping (Zeng 1993; Zeng 1994) on various computing platforms under different operating systems. All timings were done in the winter of 1999. The simulated data set has been used previously (Zeng 1994) and consists of a genetic linkage map that has four chromosomes with 16 markers on each chromosome. The markers are evenly spaced at 10 cM and the simulated data has one trait. The entire genome was scanned at a walking speed of 2 cM. The programs were run in automatic mode, with no recourse to the interactive menus. They indicate the amount of time to read in the data, perform the analysis and write the output.
Table 7.1 summarizes the timings for interval mapping.

Machine	Speed (Mhz)	Time (seconds)	Ratio to UltraSparc 60
PowerMac G3	266	7	1.7
PowerMac G3	400	5	
Sparc 10	36	34	8.5
UltraSparc 2170	167	8	2.0
UltraSparc 60	300	4	1.0
Pentium, NT 4.0	260	8	2.0
Pentium, NT 4.0	450	4	1.0

Table 7.1: Timings for Interval Mapping

Table 7.2 summarizes timings for composite interval mapping. The model for analysis was Model 6 with a window size set to 10.0 cM and using up to 5 markers to control for the genetic background. Some of the ratios for the same machine change from interval mapping to composite interval mapping. Model 6 uses quite a lot more double precision arithmetic, and this may account for the differences.
For N replications of a permutation test or bootstrap, the computing time should be less than N times the values in Tables 7.1-7.2. The jackknife analysis should be around n times these values (where n is the sample size).

Machine	Speed (Mhz)	Time (seconds)	Ratio to UltraSparc 60
PowerMac G3	266	7	1.2
Sparc 10	36	45	7.5
UltraSparc 2170	167	10	1.7
UltraSparc 60	300	6	1.0
Pentium, NT 4.0	260	10	1.7

Table 7.2: Timings for Composite Interval Mapping

Chapter 8

UNIX Man Pages

In the UNIX world, a standard way of providing online documentation of programs is to write man pages. These are ASCII text files with embedded troff commands. UNIX versions of ***QTL Cartographer*** have man pages for all the programs in the suite. On a UNIX system, if the man pages are in the correct subdirectory (in essence, if the subdirectory that contains the man pages is defined in the environmental variable MANPATH), then you can get the online help with a command such as
% man Rmap
We provide html versions of the man pages on the web server for Macintosh and Windows users. If you have World Wide Web access, first point your browser to our home page:

`http://statgen.ncsu.edu/`

Then, go down about halfway until you get to the ***QTL Cartographer*** link. Follow it to the online man pages. You can also access the rest of the ***QTL Cartographer*** manual. The manual is written in LaTeX2e and has been translated into HTML by the program **html2latex**.
The complete set of man pages are reprinted here for your benefit. Here follow the LaTeX formatted versions of the man pages.
Since the documentation will change regularly, it is a good idea to check the Web site for the current online manual. The Web pages will always be updated with the manual updates

8.1 QTLCART

NAME

QTLcart — A rudimentary front end for the QTL Cartographer system.

SYNOPSIS

QTLcart [**-h**] [**-V**] [**-A**] [**-s** *seed*] [**-W** *workdir*] [**-X** *stem*] [**-e** *logfile*] [**-R** *resource*]

DESCRIPTION

QTLcart does not actually exist. It is intended to be the front end to a set of programs collectively known as QTL Cartographer. This man page explains the options that are valid in all the programs of the QTL Cartographer suite. It also outlines how to get started using the programs.

OPTIONS

The following options can be used with any of the programs in the QTL Cartographer suite. The current programs are **Rmap**, **Rqtl**, **Rcross**, **Qstats**, **LRmapqtl**, **SRmapqtl**, **JZmapqtl**, **Eqtl**, **Prune**, and **Preplot**.

-h Prints out the current values of all program options, and information on what the program does. It then exits.

-V Turns the verbosity mode off. The programs in the suite print out messages while running. This option turns off those messages. This is useful for batch files.

-A Skips the interactive screen for setting options. All programs start up with a menu that allows setting of options. This turns the menu off. It is also very useful for batch files.

-R The programs will read the default parameters from a file specified with this option. If a file called *qtlcart.rc* is in the current working directory, it will be opened by default and all parameter values read. If no such file exists, then default parameter values will be assumed, and the file will be created.

It is probably better to simply rename a resource file *qtlcart.rc* than to use this option.

-W This option allows one to set the work directory. This directory must exist. All the input files must be in this directory and the output files will be placed there.

-s This requires a long integer to act as the random number seed. By default, it is the value returned by the ANSI C function *time()*, which is usually the number of seconds since some arbitrary past date (often 1 January 1970). This number will also be used as a unique identifer on the first line of the output file.

This can be a useful option. It is recorded in the log file when any program is run. It is possible to recreate exactly what was done using the log file.

-e This requires a filename for the log file. It will be appended to if it exists and created if not. The default is *qtlcart.log*.

-X Give a filename stem. All output will start with this stem and have extensions indicating what is in them.

EXAMPLES

For all the following examples, assume that **QTLCart** is just a wildcard for any of the programs in the suite.

```
% QTLcart -R resource.file
```

QTLcart will read option values from the file resource.file. The other programs do this, and except for **Preplot**, will regenerate the file upon exit.

```
% QTLcart -X corn
```

Will set the filename stem to *corn*. The output files will then have names beginning with *corn* and logical extensions. For example, the map file will be placed in *corn.map* and the file containing the data from a cross will be in *corn.cro*. Filenaming conventions follow the old DOS 8+3 due to historical reasons.

GLOBAL COMMAND LINE OPTIONS

All the parameters for **QTLcart** are also parameters for the other programs in the QTL Cartographer system.

GLOBAL BEHAVIOR

All the programs in the QTL Cartographer suite behave in the same general way. They were originally UNIX programs and can be run as such (using command line options). More recently, we have added an interactive menu that allows the user to set parameters. Once inside any of the programs, all the parameters of the program are displayed with their current values. The user chooses whichever parameter he or she wishes to change by selecting a number. The menu is in a loop. Choosing 0 will end the loop and proceed with the current parameter values.
The menu is also where one can get online help. Online help will be a numbered option in the list of parameters. Choose it and specify the location of the help file if the program couldn't find it.
When the programs begin to run, they will print out their parameter values to a log file (*qtlcart.log* by default).
Here is an example of the **Qstats** menu:

```
=========================================================
No.          Options                          Values:
---------------------------------------------------------
 0. Continue with these parameters
 1. Data Input File                           qtlcart.cro
 2. Output File                               qtlcart.qst
 3. Error File                                qtlcart.log
 4. Genetic Linkage Map File                  qtlcart.map
 5. Random Number Seed                        961681144
---------------------------------------------------------
 6. Specify Resource File                     qtlcart.rc
 7. Change Filename stem                      qtlcart
 8. Help
 9. Change Working Directory:
10. Quit
11. Quit, but update the Resource File
=========================================================

Please enter a number...
```

This menu is in a loop. To change a parameter, select its number and press return. You will be prompted for a new value or filename. You can clear out a filename or working directory by inputting a single period (.). When satisfied that the parameters are set correctly, you can selct 0 to run the program. If you want to quit, simply select 10. Selecting 11 will update the resource file with any parameter changes you have made.

RESOURCE FILE

The resource file keeps track of the most current parameter values used in the programs. Each time the user runs a program, the program accepts new values for parameters and writes them to the resource file. This is unlike the log file which keeps track of the parameters used at the time of running each program. The resource file that is generated by the programs in the suite is self documenting. Look in the *qtlcart.rc* file.

HELP FILE

Online help requires that **QTLcart** and all the other programs in the QTL Cartographer suite know where the helpfile is. If it is in the current working directory, there will be no problem. If not, then the user should specify the location of the help file in the resource file. The line

```
-helpfile Path+Filename
```

will allow the programs to find the helpfile. This line would look different under Windows, Macintosh and Unix systems. For Unix, a help file called *qtlcart.hlp* in the */usr/local/lib* subdirectory would be specified by

```
-helpfile   /usr/local/lib/qtlcart.hlp
```

In Windows, such a helpfile in *c:\qtlcart* would be specified by

```
-helpfile c:\qtlcart\qtlcart.hlp
```

In Macintosh, a help file on hard drive HardDrive in the folder *QTLcart* would be specified by

```
-helpfile HardDrive:QTLcart:qtlcart.hlp
```

Be aware that UNIX systems are senstive to the case of the filenames and directories, whereas Macintoshes and PCs running MS-Windows are not.
If the program can't find the helpfile, then you will be prompted for its location.

WORKING DIRECTORY

You can specify a working directory (or folder) with the **-W** option. This directory (folder) must exist prior to running any of the programs. The directory can be relative or complete, and should have the standard directory delimiter appended to it. For example

```
-W /home/user/qtlcart/work/
```

would use */home/user/qtlcart/work* as the working directory. All input and ouput files would have to be in this directory. For a Windows system, the line might be

```
-W c:\qtlcart\work\
```

whereas a Macintosh would require

```
-W HardDrive:qtlcart:work:
```

The equivalent line in the resource file would have **-workdir** instead of just **-W**.
In UNIX, you can set a path variable pointing to the programs and simply set your current working directory to the working directory. For Mac, you double click the icons and should use a working directory variable. Relative paths are also possible. For example, if the programs reside in a *bin* folder in the *qtlcart* folder on a Macintosh, then you can have a *data* folder in the *qtlcart* folder and use

```
-W ::data:
```

as the working directory. The two colons mean go up one level and then go into the *data* folder.

FILENAME STEM

The filename stem is an important concept in the usage of this package. Beginning with version 1.12, the programs utilize the filename stem *qtlcart*. All files are then named using this stem and filename extensions relevant to the filetype. For example, if the **-X** option is followed by *corn*, then when new files are created, they will have the stem *corn* followed by a logical extension. An example would be *corn.map* for a genetic linkage map. With some practice, you will be able to know the contents of a file by its extension.

USING THE INDIVIDUAL PROGRAMS

For now it is best to use the individual programs rather than the front end. If you have no data, then you would use the programs in the following order:

1. **Rmap**, to create a random map of markers.
2. **Rqtl**, to generate a random genetic model for the map.
3. **Rcross**, to create a random cross.
4. **LRmapqtl**, to do a simple linear regression of the data on the markers.
5. **SRmapqtl**, to do a stepwise linear regression of the data on the markers to rank the markers.
6. **Zmapqtl**, to do interval or composite interval mapping.
7. **Preplot**, to reformat the output of the analysis for **GNUPLOT**.
8. **GNUPLOT**, to see the results graphically.

If you have data, then you might use the programs in the following order:

1. **Rmap**, to reformat the output of MAPMAKER or a standard input file.
2. **Rcross**, to reformat your data.
3. **Qstats**, to summarize missing data and calculate some basic statistics on your quantitative traits.
4. **LRmapqtl**, to do a simple linear regression of the data on the markers.
5. **SRmapqtl**, to do a stepwise linear regression of the data on the markers to rank the markers. This should be run with model 2.
6. **Zmapqtl**, to do interval or composite interval mapping. This should be run twice, once with model 3 and a second time with model 6.
7. **Preplot**, to reformat the output of the analysis for Gnuplot.
8. **GNUPLOT**, to see the results graphically.

We recommend that the new user tries a simulation to gain an understanding of the programs

REFERENCES

1. **T. Williams and C. Kelley (1993) GNUPLOT: An Interactive Plotting Program. Version 3.5**

BUGS

Many UNIX systems have been known to get upset when trying to run the QTL Cartographer programs from out of the front end. It has something to do with the memory management. Try running the individual programs one by one. A good test is to simply run each program without changing any parameters.

SEE ALSO

Rmap(1), **Rqtl(1)**, **Rcross(1)**, **Qstats(1)**, **LRmapqtl(1)**, **SRmapqtl(1)**, **JZmapqtl(1)**, **Eqtl(1)**, **Prune(1)**, **Preplot(1)**

AUTHORS

In general, it is best to contact us via email (basten@statgen.ncsu.edu).

Christopher J. Basten, B. S. Weir and Z.-B. Zeng
Department of Statistics, North Carolina State University
Raleigh, NC 27695-8203, USA
Phone: (919)515-1934

8.2 RMAP

NAME

Rmap — Simulate or reformat a map of molecular markers

SYNOPSIS

Rmap [**-o** *output*] [**-i** *input*] [**-g** *gmode*] [**-f** *mapfunc*] [**-p** *mapparam*] [**-c** *chroms*] [**-m** *MarkersPerChrom*] [**-vm** *sdMPC*] [**-d** *InterMarkerDist*] [**-vd** *sdIMD*] [**-t** *Tails*] [**-M** *Mode*]

DESCRIPTION

Rmap creates a random map of molecular markers. The user specifies the number of chromosomes, the number of markers per chromosome and the average intermarker distance. If one specifies standard deviations for the number of markers and the average intermarker distances, they will vary subject to the normal distribution. The output gives a table of markers by chromosomes, with the distances between consecutive markers (in centiMorgans) in the table.
If you specify an input file, **Rmap** will open it, determine if it is in the same format as **Rmap** outputs, and process it based in the value given to **-g**. If the input file is the output of **MAPMAKER**, then the map will be reformatted from **MAPMAKER** into the **Rmap** output format.
Finally, there is a standard input format that **Rmap** can translate, and is defined in the file *map.inp* that comes with the distribution of the programs. Note that if the user specifies an input file, no simulations will be done and the latter half of the command line options are ignored.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.
If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This should be used with a filename indicating where the output will be written. **Rmap** will overwrite the file if it exists, and create a new file if it does not. If not used, then **Rmap** will use *qtlcart.map*.

-i You can use this option to specify an input filename. This file must exist and have one of three formats: *Rmap.out*, *map.inp* or *mapmaker.mps*. **Rmap** will attempt to identify the format of the file and translate it to another format. If you specify an input file, then the simulation parameters will be ignored.

-g Requires an integer to indicate the output format. You can use a 1 for the default ouput format, a 2 for **GNUPLOT** output or a 3 for both. If you use a 2 or a 3, then you can use **GNUPLOT** to see a primitive looking linkage map.

-f Requires an integer option to specify the mapping function. **Rmap** can use the Haldane, Kosambi, fixed or a number of other functions. The default is to use the Haldane function, which is specified with a 1. Using a 2 invokes the Kosambi mapping function. A 3 means that a fixed function is used and thus the distance in Morgans is the recombination fraction. The type of mapping function used would then be recorded in the ouput and all following analyses will use this function. One must edit the map file to change this if not using **Rmap**.

-p Requires a real number. Some map functions need an extra parameter, and this allows the user to specify it. See the manual for details.

-c This allows you to specify the number of chromosomes if you are simulating a genetic linkage map. It is 4 by default. If you are translating a file, then this will be ignored as will the remaining options.

-m This allows you to specify the average number of markers per chromosome in a simulation. The default is 16.

-vm This allows you to specify the standard deviation in the number of markers per chromosome. The number of markers per chromosome will have a normal distribution with mean given in the previous option, and the standard deviation specified here. If zero, then each chromosome will have the same number of markers.

-d Rmap uses the value given after this option as the average intermarker distance (in centiMorgans) for a simulation. It is 10 centiMorgans by default.

-vd The intermarker distance will have a normal distribution with mean set by the previous option and standard deviation specified with this option. It is 0.0 by default, which means that the intermarker distances between consecutive markers will all be the same. Set it to a positive value to have intermarker distances vary at random.

-t You can simulate maps where there are no markers on the telomeres with this option. Give this option a value of *tails* and **Rmap** puts an average of *tails* Morgans of genetic material on the ends of the chromosomes. By default, it is 0.0. If the standard deviation for intermarker distance is greater than 0.0, then then the amount of flanking DNA will have a normal distribution with mean given here and standard deviation proportional to that of the standard deviation of intermarker distances.

-M Allows you to specify an alternate simulation mode. If the **-M** option is used with a value of 1, then the intermarker distance will be used as the chromosome length (so you should make it longer), and the markers are placed on the chromosomes following the uniform distribution.

INPUT FORMAT

Rmap recognizes three types of files. The first is the *Rmap.out* format that **Rmap** itself creates. The second is a special format defined in the example file *map.inp* included in the distribution. The third format is the output of **MAPMAKER**. If the input file is a **MAPMAKER** output file, **Rmap** translates this file into its own format. If the input file is already in the correct format, **Rmap** will output it dependant upon the flag given to the **-g** option. The units of intermarker distances will be in centiMorgans in the output.

EXAMPLES

```
% Rmap -o Map.out -c 23 -vm 3 -vd 1 -t 5
```

Simulates a random map where the number of markers on each of 23 chromosomes has a normal distribution with mean 16 and standard deviation 3. The intermaker distance is normally distributed with mean 10 cM and standard deviation 1. There will be some genetic material outside the flanking markers on each chromosome, with a mean length of 5 cM and standard deviation 0.5.

```
% Rmap -o Map.out -i map.mps
```

Opens the file *map.mps*, tries to determine its format, and translates it if possible. The output will be written to the file *Map.out*. The extension **.mps** should be used with **MAPMAKER** output files and the string *-filetype mapmaker.mps* should be put somewhere in the first twenty lines of the file.

REFERENCES

1. Lander, E. S., P. Green, J. Abrahamson, A. Barlow, M. Daley, S. Lincoln and L. Newburg (1987) MAPMAKER: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics* **1**, 174−181.

2. T. Williams and C. Kelley (1993) GNUPLOT: An Interactive Plotting Program. Version 3.5

BUGS

Note that if **MAPMAKER** outputs an intermarker distance of 0.00 cM, then **Rmap** will translate it to 0.0001 cM. In fact, all intermarker distances of 0.0 will be reset to 0.0001 cM.

SEE ALSO

Rqtl(1), **Rcross(1)**, **Qstats(1)**, **LRmapqtl(1)**, **SRmapqtl(1)**, **Zmapqtl(1)**, **JZmapqtl(1)**, **Eqtl(1)**, **Prune(1)**, **Preplot(1)**, **QTLcart(1)**

AUTHORS

In general, it is best to contact us via email (basten@statgen.ncsu.edu).

Christopher J. Basten, B. S. Weir and Z.-B. Zeng
Department of Statistics, North Carolina State University
Raleigh, NC 27695-8203, USA
Phone: (919)515-1934

8.3 RQTL

NAME

Rqtl — Place a set of estimated or randomly generated QTLs on a molecular map.

SYNOPSIS

Rqtl [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-b** *beta*] [**-t** *Traits*] [**-q** *QTLperTrait*] [**-d** *dominance*] [**-1** *beta1*] [**-2** *beta2*]

DESCRIPTION

Rqtl will translate a genetic model or simulate a random model for use by **Rcross** to simulate a data set. It places a specified number of QTLs (Quantitative Trait Loci) on the molecular map created or translated by **Rmap**. For simulations, they are placed randomly on the map, and the additive and dominace effects are also determined. The molecular map could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.
If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **Rqtl** will overwrite the file if it exists, and create a new file if it does not. If not used, then **Rqtl** will use *qtlcart.qtl*.

-i This requires an input filename. This file must exist. **Rqtl** will attempt to identify the format of the file and translate it to another format. This file should contain a genetic model defining a set of QTL and including their positions and effects. See the file *qtls.inp* for the format.

-m This requires a filename that must exist. **Rqtl** will read the genetic linkage map from this file.

-t This allows the user to specify the number of traits to simulate. It is 1 by default.

-q This requires an integer argument. It allows the user to specify the number of QTL that affect the trait. If one trait is simulated, then exactly this number of QTL will be created. If more than one trait are simulated, then the number of QTL per trait will vary but have mean value specified here. The default is 9.

-d You can specify the type of dominance at the trait loci. If we assume inbred parental lines with line one marker trait alleles all Q and line two trait alleles all q, then use a 1 for no dominance, a 2 for complete dominance of Q over q, a 3 for complete dominance of q over Q, and a 4 for dominance that is random in direction and magnitude for each locus. It is 1 by default, that is no dominance.

-b Specifies the parameter needed to determine the additive effect of a QTL. It is 0.5 by default. See Zeng (1992) equation (12) and accompanying text for a discussion of this parameter. Itis not the allelic effect of a QTL allele, rather it is the shape parameter in the beta distribution.

-1, -2
Allows you to specify the two parameters used to determine the dominance effect of a QTL. The effect is simulated from a beta distribution. See the manual for more details.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**.
If a file is specified with the **-i** option, then that file will be read for the positions and effects of the QTLs. The format of this file should be identical to that of the output of **Rqtl**, or of a special format defined in the file *qtls.inp* included with the distribution.

EXAMPLES

```
% Rqtl   -d 2
```

Places 9 QTLs on the map in *Rmap.out*. There is complete dominance of A over a.

```
% Rqtl   -i qtls.inp -o test.qtl
```

Reads the file *qtls.inp* and translates it into the output format of **Rqtl**. The output is written to the file *test.qtl*, which is overwritten if it exists.

REFERENCES

1. Zeng, Zhao-Bang (1992) Correcting the bias of Wright's estimates of the number of genes affecting a quantitative trait: A further improved method. *Genetics* **132**, 823−839.

BUGS

The **-t** option for the number of traits is rather primitive at this time. The number of QTLs and their effects are randomly determined, with means given in the other options.

SEE ALSO

Rmap(1), **Rcross(1)**, **Qstats(1)**, **LRmapqtl(1)**, **SRmapqtl(1)**, **Zmapqtl(1)**, **JZmapqtl(1)**, **Eqtl(1)**, **Prune(1)**, **Preplot(1)**, **QTLcart(1)**

AUTHORS

In general, it is best to contact us via email (basten@statgen.ncsu.edu).

```
Christopher J. Basten, B. S. Weir and Z.-B. Zeng
Department of Statistics, North Carolina State University
Raleigh, NC 27695-8203, USA
Phone: (919)515-1934
```

8.4 RCROSS

NAME

Rcross — Simulate or reformat a data set.

SYNOPSIS

Rcross [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-q** *modelfile*] [**-r** *repetitions*] [**-c** *Cross*] [**-n** *SampleSize*] [**-H** *heredity*] [**-E** *Ve*] [**-I** *Interactive*]

DESCRIPTION

Rcross performs a random cross or reformats a data set. Cross types include F1 backcrosses to the P1 or P2, F2 crosses produced by selfing or random mating, recombinant inbred lines as well as a few others. It simulates marker and trait data. The markers simulated come from a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The QTL model could be a random set produced by **Rqtl** or an estimated set in the same format as the output of **Rqtl**.
Rcross can also translate files from three different formats. If the user chooses to translate a file, then the simulation options are ignored.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.
If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **Rcross** will overwrite the file if it exists, and create a new file if it does not. If not used, then **Rcross** will use *qtlcart.cro*. This output is in a format suitable for any of the mapping programs.

-i This requires an input filename. This file must exist. **Rcross** will attempt to identify the format of the file and translate it to another format. Specifying a file with this option turns off the simulation parameters below.

-m **Rcross** requires a genetic linkage map. This option require the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-q **Rcross** needs a genetic model to simulate a data set. It will read from the file specified by this option. The file specified should contain a genetic model in the same format as the output of **Rqtl**. The default file is *qtlcart.qtl*.

-H Allows the user to specify the heritability for the trait. If used, it requires a value in the range 0.0 to 1.0. It is 0.5 by default.

-E Allows the user to specify an environmental variance for the trait. If used, it requires a positive value and will disable the heritability. This is ignored by default.

-I is the flag to turn on interactive crosses. By default, it has a value of 0. To do interactive crosses, use this option with the value 1.

-c Allows the user to specify the type of cross. It requires a string such as B1, SF2 or RI1. See below for more on the values of the cross.

-n This is the sample size of the offspring. It is 200 by default and requires some integer value greater than 0 if used.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input form of the QTL data should be that of the output format from **Rqtl**. If an input file for the data is used, then it can have one of two formats. The first is identical to the *raw* files required by **MAPMAKER**. You must first use **MAPMAKER** to create a genetic map, then run the map through **Rmap** to reformat it, then use the map and the original raw file to reformat the data for subsequent use.
An alternative format is defined in a file *cross.inp* that is included with the distribution. The file can be annotated freely. Look at the *cross.inp* file and use it as a template for your data.

CROSSES

A pair of inbred parental lines (P1 and P2) that differ in the trait of interest and marker genotypes are crossed to produce an F1 generation. All crosses are then derived from these lines. Backcrossing to P1 is encoded by B1, and to P2 by B2. Selfed intercrosses of generation i are encoded by SFi. Randomly mated intercrosses of generation i are encoded by RFi. Recombinant inbreds created by selfing have the code RI1, while those by sibmating are RI2. Doubled haploids have the code RI0. A test cross of an SFi line to a Pj line is encoded by *T*(Bj)SFi. The QTL Cartographer manual explains some other crosses that are possible. Note that the UNIX shell may interpret (and) so they should either be quoted, or the cross entered into the interactive menu.

EXAMPLES

```
% Rcross  -A -V -c SF2 -n 1000
```

Does a selfed F2 cross with 1000 offspring using the linkage map in *qtlcart.map* and the model in *qtlcart.qtl*. The command line options **-A** and **-V** turn off the interactive menu and the verbosity mode, respectively.

```
% Rcross  -i cross.raw
```

Reads from the file *cross.raw*, tries to determine its format, and translates it if possible. The file *cross.raw* could be a **MAPMAKER/QTL** formatted file, a *cross.inp* formatted file or one that is already in the *Rcross.out* format.

REFERENCES

1. Lander, E. S., P. Green, J. Abrahamson, A. Barlow, M. Daley, S. Lincoln and L. Newburg (1987) MAPMAKER: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics* **1**, 174−181.

BUGS

If you use the interactive mode, you can print out the results of crosses. The analysis of these arbitrary crosses has not been fully integrated into the other programs.

SEE ALSO

Rmap(1), **Rqtl(1)**, **Qstats(1)**, **LRmapqtl(1)**, **SRmapqtl(1)**, **Zmapqtl(1)**, **JZmapqtl(1)**, **Eqtl(1)**, **Prune(1)**, **Preplot(1)**, **QTLcart(1)**

AUTHORS

In general, it is best to contact us via email (basten@statgen.ncsu.edu).

Christopher J. Basten, B. S. Weir and Z.-B. Zeng
Department of Statistics, North Carolina State University
Raleigh, NC 27695-8203, USA
Phone: (919)515-1934

8.5 PRUNE

NAME

Prune — Prune or resample the data set.

SYNOPSIS

Prune [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-I** *interactive*] [**-M** *Model*] [**-b** *simflag*]

DESCRIPTION

Prune allows one to eliminate markers or traits. It removes the data from the file containing the cross and reconstructs the molecular map. It requires a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The sample could be a randomly generated one from **Rcross** or a real one in the same format as the output of **Rcross**.
Prune also does bootstraps, permutations and simulations of missing or dominant markers.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.
If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename stem for output. **Prune** will overwrite the file ending in *.crb* if it exists, and create a new file if it does not. If not used, then **Prune** will use *qtlcart.crb*. If the map is recreated, then a new map file will be written to *qtlcart.mpb* by default or a file ending in mpb with the specified stem.

-i This requires an input filename. This file must exist. It should be in the same format as the output of **Rcross**. The default file is *qtlcart.cro*.

-m **Prune** requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-I Sets the interactive level. A zero means that Prune will do what it needs to without asking (the default for bootstraps, permutations or missing data simulations). A one means that the user will be put into a repeating loop to manipulate the data set. It has a value 1 by default, but using the **-b** option disables it.

-M This sets a level for the elimination of individuals with this much missing marker data, or for the simulation of missing or dominant markers when used with the **-b** option.

-b Prune will read in the map and data file and do one of four things depending on the value given to this option: 1. a bootstrap resampling of the data where sampling of individuals is done with replacement to create a sample of the same size as the original. 2. A permutation of the traits. A new dataset is then printed. 3. A simulation of missing markers. 4. A simulation of dominant markers. A new dataset is printed with the percent of missing marker data specified by the **-M** option. A value of zero means that this option is ignored.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**.

EXAMPLES

```
% Prune -m example.map -i example.cross -o exout
```

Puts the user into an interactive menu for eliminating traits, markers, etc.

```
% Prune -m example.map -i example.cross -o exout -b 1
```

The **-b** option creates a new sample from the old. The new sample is created by resampling the original sample with replacement. Phenotypes and genotypes are kept together. The new sample will have the same sample size as the old one. It will be written to *exout.crb*. No new map will be written.

REFERENCES

BUGS

You can eliminate multiple markers in the interactive loop. You should be aware that the order marker elimination is important. If all the markers to be eliminated are on separate chromosomes, the order is unimportant. If two markers from the same chromosome are to be eliminated, order should be to eliminate the highest numbered marker. The same concept holds for traits: eliminate them in the order of highest to lowest.
Do not try to eliminate any markers or traits AND do a bootstrap, permutation or simulation of missing markers in the same run.

SEE ALSO

Rmap(1), **Rqtl(1)**, **Rcross(1)**, **Qstats(1)**, **LRmapqtl(1)**, **SRmapqtl(1)**, **JZmapqtl(1)**, **Eqtl(1)**, **Preplot(1)**, **QTLcart(1)**

AUTHORS

In general, it is best to contact us via email (basten@statgen.ncsu.edu).

Christopher J. Basten, B. S. Weir and Z.-B. Zeng
Department of Statistics, North Carolina State University
Raleigh, NC 27695-8203, USA
Phone: (919)515-1934

8.6 QSTATS

NAME

Qstats — Calculate basic statistics for a QTL dataset.

SYNOPSIS

Qstats [**-o** *output*] [**-i** *input*] [**-m** *mapfile*]

DESCRIPTION

Qstats does some basic statistics on a dataset of quantitative traits. It plots a histogram and calculates the sample size, mean, variance standard deviation, skewness, kurtosis, and average deviation for a quantitative trait. The program also summarizes missing marker and trait data, as will as determining the marker types (dominant or codominant). Finally, **Qstats** will test whether markers are segregating at random. It requires a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The sample could be a randomly generated one from **Rcross** or a real one in the same format as the output of **Rcross**.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.
If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **Qstats** will append the file if it exists, and create a new file if it does not. If not used, then **Qstats** will use *qtlcart.qst*.

-i This requires an input filename. This file must exist. It should be in the same format as the output of **Rcross**. The default file is *qtlcart.cro*.

-m **Qstats** requires a genetic linkage map. This option require the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**.

EXAMPLES

```
% Qstats -i corn.cro -m corn.map
```

Calculates basic statistics on the dataset in *corn.cro* using the genetic linkage map in *corn.map*. The program will display and interactive menu for setting options and print out messages to the screen while running. These can be turned off with **-A** and **-V**, respectively. If the dataset in *corn.cro* has more than one trait, then all traits will be analyzed.

REFERENCES

1. M. Lynch and B. Walsh (1998) Genetics and Analysis of Quantitative Traits. Sinauer Associates, Sunderland, MA.

BUGS

Are there any other statistics that we can do? Your suggestions are welcome.

SEE ALSO

Rmap(1), **Rqtl(1)**, **Rcross(1)**, **LRmapqtl(1)**, **SRmapqtl(1)**, **Zmapqtl(1)**, **JZmapqtl(1)**, **Eqtl(1)**, **Prune(1)**, **Preplot(1)**, **QTLcart(1)**

AUTHORS

In general, it is best to contact us via email (basten@statgen.ncsu.edu).

Christopher J. Basten, B. S. Weir and Z.-B. Zeng
Department of Statistics, North Carolina State University
Raleigh, NC 27695-8203, USA
Phone: (919)515-1934

8.7 LRMAPQTL

NAME

LRmapqtl — Single marker QTL analysis.

SYNOPSIS

LRmapqtl [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-r** *reps*] [**-t** *trait*]

DESCRIPTION

LRmapqtl uses simple linear regression to map quantitative trait loci to a map of molecular markers. It requires a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The sample could be a randomly generated one from **Rcross** or a real one in the same format as the output of **Rcross**.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.
If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **LRmapqtl** will append the file if it exists, and create a new file if it does not. If not used, then **LRmapqtl** will use *qtlcart.lr*.

-i This requires an input filename. This file must exist. It should be in the same format as the output of **Rcross**. The default file is *qtlcart.cro*.

-m LRmapqtl requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-r LRmapqtl will do a permutation test *a la* Churchill and Doerge (1994). This option specifies the number of permutions to do. It is zero by default, which means no permuation test is done. If used, you must specify a positive integer. Usually, 1,000 is sufficient.

-t Use this to specify which trait **LRmapqtl** will analyze. If this number is greater than the number of traits, then all traits will be analyzed. The default is to analyze trait 1 only.

MODEL

The basic linear model is

```
Trait = Mean + Slope x Marker + Error
```

The marker value will be in the range [-1, 1] inclusive. Two hypotheses are compared. The null hypothesis is that the Slope is zero. The alternate is that the Slope is non-zero. A p-value for the likelihood ratio of these to hypotheses is calculated for each marker-trait combination. **LRmapqtl** outputs a table with parameter estimates, F statistics, Likelihood ratios and p-values.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**.

EXAMPLES

```
% LRmapqtl -i corn.cro -m corn.map
```

Calculates the regression coefficients for each marker on the dataset in *corn.cro* using the genetic linkage map in *corn.map*.

REFERENCES

1. Churchill, G. A. and R. W. Doerge (1994) Empirical threshold values for quantitative trait mapping. *Genetics* **138**, 963−971.

BUGS

SEE ALSO

Rmap(1), **Rqtl(1)**, **Rcross(1)**, **Qstats(1)**, **SRmapqtl(1)**, **Zmapqtl(1)**, **JZmapqtl(1)**, **Eqtl(1)**, **Prune(1)**, **Preplot(1)**, **QTLcart(1)**

AUTHORS

In general, it is best to contact us via email (basten@statgen.ncsu.edu).

```
Christopher J. Basten, B. S. Weir and Z.-B. Zeng
Department of Statistics, North Carolina State University
Raleigh, NC 27695-8203, USA
Phone: (919)515-1934
```

8.8 SRMAPQTL

NAME

SRmapqtl — Map quantitative traits on a molecular map.

SYNOPSIS

SRmapqtl [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-t** *trait*] [**-M** *Model*] [**-F** *pFin*] [**-B** *pFout*]

DESCRIPTION

SRmapqtl uses stepwise regression to map quantitative trait loci to a map of molecular markers. It requires a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The sample could be a randomly generated one from **Rcross** or a real one in the same format as the output of **Rcross**.
This program should be run before .**Zmapqtl** if you want to use composite interval mapping. The results will be used to pick markers background control in composite interval mapping. The main result from using this program is to rank the markers in terms of their influence on the trait of interest.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.
If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **SRmapqtl** will append the file if it exists, and create a new file if it does not. If not used, then **SRmapqtl** will use *qtlcart.sr*.

-i This requires an input filename. This file must exist. It should be in the same format as the output of **Rcross**. The default file is *qtlcart.cro*.

-m **SRmapqtl** requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-t Use this to specify which trait **SRmapqtl** will analyze. If this number is greater than the number of traits, then all traits will be analyzed. The default is to analyze trait 1 only.

-M This tells **SRmapqtl** what type of analysis to perform. Use a 0 for forward stepwise (FS) regression, a 1 for backward elimination (BE) and a 2 for forward regression with a backward elimination step at the end (FB). It is probably best to use Model 2 here.

-F Requires a real number in the range 0.0 to 1.0. This is a threshold p value for adding markers in model 2 during the forward stepwise regression step. The default is 0.05.

-B Requires a real number in the range 0.0 to 1.0. This is a threshold p value for deleting markers in model 2 during the backward elimination step. It should probably be the same as the previous option. The default is 0.05.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**.

EXAMPLES

```
% SRmapqtl -i corn.cro -m corn.map -M 2
```

Does a forward stepwise regression with a backward elimination step for the dataset in *corn.cro* using the genetic linkage map in *corn.map*.

REFERENCES

BUGS

Forward and backward regression should probably use the thresholds for adding and deleting markers from the model. When that feature is added, the **-F** and **-B** options will have more use.

SEE ALSO

Rmap(1), **Rqtl(1)**, **Rcross(1)**, **Qstats(1)**, **LRmapqtl(1)**, **Zmapqtl(1)**, **JZmapqtl(1)**, **Eqtl(1)**, **Prune(1)**, **Preplot(1)**, **QTLcart(1)**

AUTHORS

In general, it is best to contact us via email (basten@statgen.ncsu.edu).

Christopher J. Basten, B. S. Weir and Z.-B. Zeng
Department of Statistics, North Carolina State University
Raleigh, NC 27695-8203, USA
Phone: (919)515-1934

8.9 ZMAPQTL

NAME

Zmapqtl — Composite interval mapping module

SYNOPSIS

Zmapqtl [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-l** *lrfile*] [**-S** *srfile*] [**-t** *trait*] [**-M** *Model*] [**-c** *chrom*] [**-d** *walk*] [**-n** *nbp*] [**-w** *window*] [**-r** *perms*] [**-r** *boots*]

DESCRIPTION

Zmapqtl uses composite interval mapping to map quantitative trait loci to a map of molecular markers. It requires a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The sample could be a randomly generated one from **Rcross** or a real one in the same format as the output of **Rcross**. In addition, the program requires the results of the stepwise linear regression analysis of **SRmapqtl** for composite interval mapping.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.
If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **Zmapqtl** will append the file if it exists, and create a new file if it does not. If not used, then **Zmapqtl** will use *qtlcart.z*.

-i This requires an input filename. This file must exist. It should be in the same format as the output of **Rcross**. The default file is *qtlcart.cro*.

-m **Zmapqtl** requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-t Use this to specify which trait **Zmapqtl** will analyze. If this number is greater than the number of traits, then all traits will be analyzed. The default is to analyze trait 1 only.

-l Allows the user to specify the name of the file containing results from **LRmapqtl**. **Zmapqtl** reads those results and uses the information to choose cofactors for some of the analysis methods.

-S Allows the user to specify the name of the file containing results from **SRmapqtl**. **Zmapqtl** reads the results and uses the information to choose cofactors for composite interval mapping model 6.

-M Zmapqtl assumes the specified model (see below) in the analysis. Model 3 is default.

-c The user can specify a specific chromosome for **Zmapqtl** to analyze. If zero, then all will be analyzed.

-d Zmapqtl walks along the chromosome at a rate that can be specified with this option. The default is to do an analysis every 2 centiMorgans along the chromosome.

-n Use this to indicate how many background parameters **Zmapqtl** uses in composite interval mapping. This is used only with model 6, and gives an upper bound. If fewer than this number of markers are ranked in the *SRmapqtl.out* file, then less than the specified number of markers will be used.

-w Zmapqtl blocks out a region of this many centiMorgans on either side of the markers flanking the test position when picking background markers. It is 10 by default and is only used in models 5 and 6. We refer to it as the *window size*.

-r Zmapqtl can do a permutation test to determine the threshold for rejecting the null hypothesis of no QTL at a site. By default, this option sets the number of permutations equal to 0, which means no permutation test is run. You can set it to a number $<$ 10000 to do the test. See Churchill and Doerge (1994) for more details. The results are in an interim file. Use **Eqtl** to summarize them when enough repetitions have been done. You need to run **Zmapqtl** without permutations or bootstraps at least once before you can do the permutation tests. This option only allows for interval mapping (Model 3) or composite interval mapping (Model 6).

-b When used with argument 1, **Zmapqtl** will do a single bootstrap. You need to run **Prune** to actually create the bootstrapped data set: This option merely analyzes it and stores summary statistics in an interim file *qtlcart.z3b* by default, for model 3. You should also run **Zmapqtl** without bootstraps or permutation tests before doing a bootstrap analysis.

When used with an argument 2, **Zmapqtl** will do a jackknife analysis. Again, **Zmapqtl** should be run without this argument prior to doing a jackknife.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**.

EXAMPLES

```
% Zmapqtl
```

Calculates the likelihood ratio test statistics of the dataset in *qtlcart.cro* using the map in *qtlcart.map*.

```
% nice Zmapqtl -A -V -i corn.cro -m corn.map -M 6 -r 500 &
```

Calculates the likelihood ratio test statistics of the dataset in *corn.cro* using the map in *corn.map*. Model 6 is used for analysis and a permutation test with 500 replications is performed. The program is nice'd as a courtesy to other users, and run in the background so that the user can logout and relax.

MODELS

Different parameters for the **-M** option allow for the analysis of the data assuming different models. Models 1−3 were described in Zeng (1993, 1994).

1. Fit all the background markers
2. Fit all unlinked background markers
3. Fit only the mean (Lander and Botstein (1989) method)
4. Fit a subset of the other markers, namely those unlinked markers with the highest correlation with the trait on each chromosome.
5. This model uses a pair of markers from each other chromosome and all linked markers that fall outside a window around the flanking markers. This window extends to 10 cM beyond the markers immediately flanking the test position. The window size can be changed with the **-w** option.
6. This model uses a specified number of markers that fall outside a window around the flanking markers. This window extends to 10 cM beyond the markers immediately flanking the test position. The number of markers are set by the **-n** option. You need to run **SRmapqtl** to rank the markers before using model 6.

The default is to fit only the mean, that is to use interval mapping.

PERMUTATION TESTS

Churchill and Doerge (1994) describe a method to calculate the threshold values for quantitative trait mapping that we have implemented in this program. Basically, it does a permutation of the trait values and the genotypes and redoes the analysis. Over the number of replicates, two types of thresholds are defined: "experimentwise" and "comparisonwise". We calculate the experimentwise thresholds, but only give p values for the comparisonwise values to save on storage space. The p values give the proportion of permuted replicates that have loglikelihood ratios larger than the observed ratios.

If you choose to do permutation tests, you need to run **Zmapqtl** with the model of choice prior to doing the permutation test. Also, if the program terminates prematurely, you can restart it from where it left off to complete the permutation test.

REFERENCES

1. Churchill, G. A. and R. W. Doerge (1994) Empirical threshold values for quantitative trait mapping. *Genetics* **138**, 963−971.

2. Lander, E. S. and D. Botstein (1989) Mapping Mendelian factors underlying quantitative traits using RFLP linkage maps. *Genetics* **121**, 185−199.

3. Zeng, Zhao-Bang (1993) Theoretical basis for separation of multiple linked gene effects in mapping quantitative trait loci. *Proc. Natl. Acad. Sci., USA* **90**, 10972−10976.

4. Zeng, Zhao-Bang (1994) Precision mapping of quantitative trait loci. *Genetics* **136**, 1457−1468.

BUGS

It is likely that we will abandon the internal permutation tests in **Zmapqtl**. It is more efficient to use **Prune** and a batch file to do the same job. This paradigm will allow users to do permutation tests with any of the programs.

SEE ALSO

Rmap(1), **Rqtl(1)**, **Rcross(1)**, **Qstats(1)**, **LRmapqtl(1)**, **SRmapqtl(1)**, **JZmapqtl(1)**, **Eqtl(1)**, **Prune(1)**, **Preplot(1)**, **QTLcart(1)**

AUTHORS

In general, it is best to contact us via email (basten@statgen.ncsu.edu).

```
Christopher J. Basten, B. S. Weir and Z.-B. Zeng
Department of Statistics, North Carolina State University
Raleigh, NC 27695-8203, USA
Phone: (919)515-1934
```

8.10 JZMAPQTL

NAME

JZmapqtl — Multitrait mapping module

SYNOPSIS

JZmapqtl [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-E** *eqtfile*] [**-S** *srfile*] [**-t** *trait*] [**-M** *Model*] [**-c** *chrom*] [**-d** *walk*] [**-n** *nbp*] [**-w** *window*] [**-I** *hypo*]

DESCRIPTION

JZmapqtl uses (composite) interval mapping to map quantitative trait loci to a map of molecular markers and can analyze multiple traits simultaneously. It requires a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The sample could be a randomly generated one from **Rcross** or a real one in the same format as the output of **Rcross**. In addition, the program requires the results of the stepwise linear regression analysis of **SRmapqtl** for composite interval mapping.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.
If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **JZmapqtl** will append the file if it exists, and create a new file if it does not. If not used, then **JZmapqtl** will use *qtlcart.zj*, where the *j* indicates the trait analyzed and the zero'th file contains joint mapping.

-i This requires an input filename. This file must exist. It should be in the same format as the output of **Rcross**. The default file is *qtlcart.cro*.

-m **JZmapqtl** requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-t Use this to specify which trait **JZmapqtl** will analyze. If this number is greater than the number of traits, then all traits will be analyzed unless the trait name begins with a minus sign. If a negative number is given, then only traits beginning with a plus sign will be analyzed. The default is to analyze trait 1 only.

-E Allows the user to specify the name of the file containing results from **Eqtl**. **JZmapqtl** reads those results and uses the information to choose cofactors for some of the analysis methods.

-S Allows the user to specify the name of the file containing results from **SRmapqtl**. **JZmapqtl** reads the results and uses the information to choose cofactors for composite interval mapping model 6.

-M JZmapqtl assumes the specified model (see below) in the analysis. Model 3 is default.

-c The user can specify a specific chromosome for **Zmapqtl** to analyze. If zero, then all will be analyzed.

-d Zmapqtl walks along the chromosome at this rate. The default is to do an analysis every 2 centiMorgans along the chromosome.

-n Use this to indicate how many background parameters **JZmapqtl** uses in composite interval mapping. This is used only with model 6, and gives an upper bound. If fewer than this number of markers are ranked in the *SRmapqtl.out* file, then less than the specified number of markers will be used.

-w JZmapqtl blocks out a region of this many centiMorgans on either side of the markers flanking the test position when picking background markers. It is 10 by default and is only used in models 5 and 6. We refer to it as the *window size*.

-I JZmapqtl requires the user to specify which hypotheses to test. For backcrosses, there are two hypotheses numbered 1 and 0. Use 10 for backcrosses or a 14 to do GxE tests as well. For crosses in which there are three genotypic classes, there are hypotheses 0, 1, 2, and 3. Use 30, 31, 32 in that case or 34 to do GxE. These are explained in greater detail in the manual.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**.

EXAMPLES

```
% JZmapqtl
```

Calculates the likelihood ratio test statistics of the dataset in *qtlcart.cro* using the map in *qtlcart.map*.

```
% nice JZmapqtl -A -V -i corn.cro -m corn.map -M 6 -t 3 -I 34 &
```

Calculates the likelihood ratio test statistics of the dataset in *corn.cro* using the map in *corn.map*. Model 6 is used for analysis. This file has two traits, so specifying trait 3 means that both traits are analyzed. Hypothesis 34 means that GxE interactions are also analyzed. The program is nice'd as a courtesy to other users, and run in the background so that the user can logout and relax.

MODELS

Different parameters for the **-M** option allow for the analysis of the data assuming different models. See the **Zmapqtl** man page for explanations of models 3, 6 and 7. These are the only models available in **JZmapqtl**

REFERENCES

1. Lander, E. S. and D. Botstein (1989) Mapping Mendelian factors underlying quantitative traits using RFLP linkage maps. *Genetics* **121**, 185−199.

2. Zeng, Zhao-Bang (1993) Theoretical basis for separation of multiple linked gene effects in mapping quantitative trait loci. *Proc. Natl. Acad. Sci., USA* **90**, 10972−10976.

3. Zeng, Zhao-Bang (1994) Precision mapping of quantitative trait loci. *Genetics* **136**, 1457−1468.

4. Jiang, Changjian and Zhao-Bang Zeng (1995) Multiple trait analysis of genetic mapping for quantitative trait loci. *Genetics* **140**, 1111−1127.

BUGS

Preplot ignores the output at present. So far, the program only does joint mapping and one form of GxE. Tests for close linkage, pleiotopic effects and other environmental effects will be added in the future.

HINTS

You can select traits to include in the analysis in three ways:

a. Set the trait to analyze at 0, so that no traits except those beginning with a **[+]** (plus sign) are analyzed. You would need to edit the .cro file first to prepend a **+** to all traits you wanted in the analysis.

b. Set the trait to a value in the range **[1-t]** inclusive, where t is the number of traits in the .cro file. You will then get single trait results.

c. Set the trait to a value greater than t. Then all traits will be put in the analysis, unless they begin with a minus sign **[-]**. As in a. above, you would need to edit the .cro file to minus out some traits.

You need to set the hypothesis test for SFx and RFx crosses. The default of 10 is ok for crosses in which there are only two marker genotypic classes (BCx, RIx). To test GxE, use 14. For SFx and RFx, values of 30, 31 or 32 are valid, and a 34 invokes the GxE test. Recall that we have the following hypotheses:

1. H0: a = d = 0
2. H1: a !=0 , d = 0
3. H2: a = 0 , d != 0
4. H3: a != 0, d != 0

For 30, we test H3:H0. For 31, we test H3:H0, H3:H1 and H1:H0. For 32, we test H3:H0, H3:H2 and H2:H0. 30 is probably fine for initial scans. Hypothesis 34 does a test for H3:H0 as well as the GxE.
For Model 6, be sure to run **SRmapqtl** first. Once done, **JZmapqtl** will use all markers that are significant for any of the traits in the analysis. We need to work out a better way to select the cofactors. Presently we use any markers that are significant for any trait. Also, be sure to use FB regression (Model 2 in **SRmapqtl**), or else you will end up using all markers as cofactors.

SEE ALSO

Rmap(1), **Rqtl(1)**, **Rcross(1)**, **Qstats(1)**, **LRmapqtl(1)**, **SRmapqtl(1)**, **Zmapqtl(1)**, **Eqtl(1)**, **Prune(1)**, **Preplot(1)**, **QTLcart(1)**

AUTHORS

In general, it is best to contact us via email (basten@statgen.ncsu.edu).

Christopher J. Basten, B. S. Weir and Z.-B. Zeng
Department of Statistics, North Carolina State University
Raleigh, NC 27695-8203, USA
Phone: (919)515-1934

8.11 PREPLOT

NAME

Preplot — Process results of LRmapqtl and Zmapqtl for input to gnuplot

SYNOPSIS

Preplot [**-o** *output*] [**-m** *mapfile*] [**-l** *lrfile*] [**-z** *zfile*] [**-q** *qtlfile*] [**-S** *threshold*] [**-T** *terminal*] [**-H** *hypo*] [**-L** *lod*]

DESCRIPTION

Preplot reformats the output of **LRmapqtl** and **Zmapqtl** so that it can be plotted by **GNUPLOT**. It requires a molecular map that was used in the analysis of the data with **LRmapqtl** and **Zmapqtl**.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.
If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename stem for output. **Preplot** will overwrite the file if it exists, and create a new file if it does not. If not used, then **Preplot** will use *qtlcart*. The **GNUPLOT** file will be *qtlcart.plt* in that case.

-m **Zmapqtl** requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-l This requires an input filename. This file must exist. It should be in the same format as the output of **LRmapqtl**. The default file is *qtlcart.lr*.

-q This requires an input filename. This file may or may not exist. It should be in the same format as the output of **Rqtl**. The default file is *qtlcart.qtl*.

-z This requires an input filename. This file must exist. It should be in the same format as the output of **Zmapqtl**. The default file is *qtlcart.z*.

-T Allows the user to set the output terminal. Valid options can be found in the **GNUPLOT** manual. The default is *x11* on UNIX, *mac* for Macintosh and *windows* for MS-Windows.

-S When given an argument, **Preplot** will use this significance threshold. It is 3.84 by default.

-H **Preplot** will get results for this hypothesis test from the **Zmapqtl** outputfile. Test 1 is the default, which is the only value for a backcross.

-L If given an argument of 1, **Preplot** will output LOD scores instead of the LR test statistics.

EXAMPLES

```
% Preplot -L 1
```

Preplot will automagically reformat your results to be plotted by **GNUPLOT**, converting the likelihood ratio test statistics into LOD scores along the way.

REFERENCES

1. T. Williams and C. Kelley (1993) GNUPLOT: An Interactive Plotting Program. Version 3.5

BUGS

Preplot ignores **JZmapqtl** output.

SEE ALSO

Rmap(1), **Rqtl(1)**, **Rcross(1)**, **Qstats(1)**, **LRmapqtl(1)**, **SRmapqtl(1)**, **JZmapqtl(1)**, **Eqtl(1)**, **Prune(1)**, **Zmapqtl(1)**, **QTLcart(1)**

AUTHORS

In general, it is best to contact us via email (basten@statgen.ncsu.edu).

```
Christopher J. Basten, B. S. Weir and Z.-B. Zeng
Department of Statistics, North Carolina State University
Raleigh, NC 27695-8203, USA
Phone: (919)515-1934
```

8.12 EQTL

NAME

Eqtl — Summarize the output of **Zmapqtl**

SYNOPSIS

Eqtl [**-o** *output*] [**-z** *zmapfile*] [**-m** *mapfile*] [**-t** *trait*] [**-M** *Model*] [**-a** *size*] [**-S** *threshold*] [**-L** *lod*]

DESCRIPTION

Eqtl reformats the prodigous output of **Zmapqtl**. The output file has a section that is suitable for input to **Rcross**. There are other sections to the output that are more readable. **Eqtl** can also detect whether a bootstrap, permutation or jackknife analysis was performed and process the interim files produced by those analyses.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.
If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **Eqtl** will overwrite the file if it exists, and create a new file if it does not. If not used, then **Eqtl** will use *qtlcart.eqt*.

-z This requires an input filename. This file must exist. It should be in the same format as the output of **Zmapqtl**. The default file is *qtlcart.z*.

-m **Eqtl** requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-H Allows the user to specify which hypothesis test results to process. Use values 10 or 14 for data with two marker classes, and 30, 31, 32, 34 for those with three marker classes.

-S Tells **Eqtl** the significance threshold. It assumes that the test statistic is significant if greater than this value. It is 3.84 by default.

-a **Eqtl** uses the specified size (alpha) to determine the significance threshold from the experiment-wise permutation results. If used, the **-S** option is ignored, and the significance threshold is set and saved from the experiment-wise permutation test results. The size is 0.05 by default.

-L If used with argument 1, it causes LOD scores to be output rather than the LR statistics. It is 0 by default.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**. The other files should have been created by **Zmapqtl**. Take care that **Zmapqtl** completed its analysis: An incomplete *qtlcart.z* file can cause **Eqtl** to crash.

EXAMPLES

```
% Eqtl -m example.map -z example.z -S 13.2
```

reprocesses the results of example.z based on the map in example.map using a significance threshold of 13.2.

BOOTSTRAPS, JACKKNIVES AND PERMUTATIONS

If **Zmapqtl** was used to do a bootstrap experiment or a permutation test, then there will be interim results files. With the default filename stem and model 3, there will be files *qtlcart.z3c* and *qtlcart.z3e* if a permutation test was done, and *qtlcart.z3a* if a bootstrap was done. **Eqtl** automatically detects these files and processes their results. It will open a qtlcart.z3e file and determine an experimentwise threshold based on the size specified with the **-a** option.
If the *qtlcart.z3a* file exists, then **Eqtl** opens it and computes the means and standard deviations, at each test site, of the likelihood ratio test statistic, additive effect and dominance effect. The results are printed to *qtlcart.z3b*.
The jackknife procedure produces a *qtlcart.z3i* that **Eqtl** opens computes the means and standard deviations, at each test site of the likelihood ratio test statistic, additive effect and dominance effect. The results are printed to *qtlcart.z3j*.

REFERENCES

BUGS

If the resource file indicates that there are more than one trait, then Eqtl will try to estimate positions and additive effects for all the traits. This will even if no analysis was done on the extra traits. The output file will then have some null estimates.
When doing a jackknife with **Zmapqtl**, the user should check that the file ending in the letter i is truely the last version of the interim jackknife file. **Zmapqtl** switches between a file ending in i and another ending in j, so check both and move the j file onto the i file if required.
If you set the significance threshold too high, then **Eqtl** may find no QTL in the *qtlcart.z* output. If this is the case, then **Eqtl** will crash.

SEE ALSO

Rmap(1), **Rqtl(1)**, **Rcross(1)**, **Qstats(1)**, **LRmapqtl(1)**, **SRmapqtl(1)**, **JZmapqtl(1)**, **Zmapqtl(1)**, **Prune(1)**, **Preplot(1)**, **QTLcart(1)**

AUTHORS

In general, it is best to contact us via email (basten@statgen.ncsu.edu).

Christopher J. Basten, B. S. Weir and Z.-B. Zeng
Department of Statistics, North Carolina State University
Raleigh, NC 27695-8203, USA
Phone: (919)515-1934

Bibliography

Basten, C. J., B. S. Weir, and Z.-B. Zeng (1994). Zmap–a QTL cartographer. In C. Smith, J. S. Gavora, B. B. J. Chesnais, W. Fairfull, J. P. Gibson, B. W. Kennedy, and E. B. Burnside (Eds.), *Proceedings of the 5th World Congress on Genetics Applied to Livestock Production: Computing Strategies and Software*, Volume 22, Guelph, Ontario, Canada, pp. 65–66. Organizing Committee, 5th World Congress on Genetics Applied to Livestock Production.

Carter, T. C. and D. S. Falconer (1951). Stocks for detecting linkage in the mouse and the theory of their design. *J. Genet. 50*, 307–323.

Churchill, G. A. and R. W. Doerge (1994). Empirical threshold values for quantitative trait mapping. *Genetics 138*, 963–971.

Cockerham, C. C. and Z. Zeng (1996). Design III with marker loci. *Genetics 143*, 1437–1456.

Doerge, R. W. and G. A. Churchill (1996). Permutation tests for multiple loci affecting a quantitative character. *Genetics 142*, 285–294.

Doerge, R. W., Z. Zeng, and B. S. Weir (1997). Statistical issues in the search for genes affecting quantitative traits in experimental populations. *Stat. Sci. 0*, 000–000.

Dongarra, J. J., C. B. Moler, J. R. Bunch, and G. W. Stewart (1979). *LINPACK Users' Guide*. Philadelphia, PA: SIAM.

Falconer, D. S. and T. F. C. MacKay (1996). *Introduction to Quantitative Genetics*. Essex, UK: Longman Group Limited.

Felsenstein, J. (1979). A mathematically tractable family of genetic mapping functions with different amounts of interference. *Genetics 91*, 769–775.

Fisch, R. D., M. Ragot, and G. Gay (1996). A generalization of the mixture model in the mapping of quantitative trait loci for progeny from a bi-parental cross of inbred lines. *Genetics 143*, 571–577.

Haldane, J. B. S. (1919). The combination of linkage values and the calculation of distances between the loci of linked factors. *J. Genet. 8*, 299–309.

Horvat, S. and J. F. Medrano (1995). Interval mapping of *high growth (hg)*, a major locus that increases weight gain in mice. *Genetics 139*, 1737–1748.

Jiang, C. and Z. Zeng (1995). Multiple trait analysis of genetic mapping for quantitative trait loci. *Genetics 140*, 1111–1127.

Jiang, C. and Z. Zeng (1997). Mapping quantitative trait loci with dominant and missing markers in various crosses from two inbred lines. *Genetica 101*, 47–58.

Karlin, S. (1984). Theoretical aspects of genetic map functions in recombination processes. In A. Chakravarti (Ed.), *Human Population Genetics: The Pittsburgh Symposium*, New York, pp. 209–228. Van Nostrand Reinhold.

Kosambi, D. D. (1944). The estimation of map distances from recombination values. *Ann. Eugen. 12*, 172–175.

Lander, E. S. and D. Botstein (1989). Mapping mendelian factors underlying quantitative traits using rflp linkage maps. *Genetics 121*, 185–199.

Lander, E. S., P. Green, J. Abrahamson, A. Barlow, M. Daley, S. Lincoln, and L. Newburg (1987). MAPMAKER: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics 1*, 174–181.

Lincoln, S., M. Daly, and E. S. Lander (1992). Constructing genetic maps with MAPMAKER/EXP 3.0. Technical report, Whitehead Institute Technical Report.

Liu, B. (1998). *Statistical Genomics: Linkage, Mapping and QTL Analysis*. Boca Raton, FL: CRC PRess LLC.

Lynch, M. and B. Walsh (1998). *Genetics and Analysis of Quantitative Traits*. Sunderland, MA: Sinauer Associates, Inc.

Meng, X. and D. B. Rubin (1993). Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika 80*, 267–268.

Morgan, T. H. (1994). *The Theory of Genes*. New Haven, CN: Yale University Press.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling (1988). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge, UK: Cambridge University Press.

Rao, D. C., B. J. Keats, J. M. Lalouel, N. E. Morton, and S. Lee (1979). A maximum likelihood map of chromosome 1. *A. J. Hum. Genet. 31*, 680–696.

Sturt, E. (1976). A mapping function for human chromosomes. *Ann. Hum. Genet., Lond. 40*, 147–147.

Williams, T. and C. Kelley (1993). *GNUPLOT: An Interactive Plotting Program*. Version 3.5.

Zeng, Z. (1992). Correcting the bias of wright's estimates of the number of genes affecting a quantitative trait: A further improved method. *Genetics 131*, 987–1001.

Zeng, Z. (1993). Theoretical basis for separation of multiple linked gene effects in mapping quantitative trait loci. *Proc. Natl. Acad. Sci. USA 90*, 10972–10976.

Zeng, Z. (1994). Precision mapping of quantitative trait loci. *Genetics 136*, 1457–1468.

Index

-A, 96
-a, 131
-B, 120
-b, 107, 113, 122
-c, 103, 110, 122, 126
-d, 103, 107, 122, 126
-E, 110, 126
-e, 97
-F, 120
-f, 103
-g, 103
-H, 110, 130, 131
-h, 96
-I, 110, 112, 126
-i, 102, 106, 109, 112, 115, 117, 119, 121, 125
-L, 130, 132
-l, 121, 129
-M, 103, 113, 119, 122, 126
-m, 103, 106, 109, 112, 115, 117, 119, 121, 125, 129, 131
-n, 110, 122, 126
-o, 102, 106, 109, 112, 115, 117, 119, 121, 125, 129, 131
-p, 103
-q, 106, 109, 129
-R, 96
-r, 117, 122
-S, 122, 126, 130, 131
-s, 96
-T, 129
-t, 103, 106, 117, 119, 121, 125
-V, 96
-vd, 103
-vm, 103
-W, 96
-w, 122, 126
-X, 97
-z, 129, 131

T. Williams and C. Kelley (1993) GNUPLOT: An Interactive Plotting Program. Version 3.5, 100

a., 127
additive effect, 37, 39

b., 127
background parameter, 58
beta distribution, 37
bootstrap, 27, 44, 60, 70
BOOTSTRAPS, JACKKNIVES AND PERMUTATIONS, 132
bug, 22

c., 127
categorical trait, 42, 53
command line, 23
composite interval mapping, 46, 56, 65, 93
covariate, 47, 69
cross
  advanced intercross, 14
  backcross, 14, 62
  Design III, 14
  doubled haploid, 14
  intercross, 14, 63
  recombinant inbred line, 14
  repeated backcross, 14
  test cross, 14
CROSSES, 110

dominance, 37, 43

ECM algorithm, 57

Eqtl, 45, 47, 64, 69
EQTL, 131
Eqtl
    options, 71
    output, 69
experimentwise significance level, 65
experimentwise significance threshold, 47

filename stem, 31
FILENAME STEM, 99
ftp server, 19, 75, 81

gamma distribution, 37
gamma function, 37
genetic linkage map, 14, 16, 33, 38
genotype, 39
GLOBAL BEHAVIOR, 97
GLOBAL COMMAND LINE OPTIONS, 97
GNUPLOT, 17, 33, 36, 72, 73

help file, 27
HELP FILE, 98
heritability, 38–40
HINTS, 127

inbred line, 14
INPUT FORMAT, 104, 107, 110, 113, 115, 118, 120, 122, 126, 132
install
    Macintosh, 21
    MS-Windows, 20
    UNIX, 20
interactive menu, 26, 27, 42
interval mapping, 46, 56, 65, 93

jackknife, 60, 70
JZmapqtl, 65
JZMAPQTL, 125
JZmapqtl
    option, 65

least squares, 54
linear regression, 53
LINPACK, 18
LOD, 72
log file, 27
LR, 72
LRmapqtl, 53
LRMAPQTL, 117
LRmapqtl
    options, 54
    output, 54

Macintosh, 16, 19, 24, 44
    binhex, 21
    install, 21
    StuffitExpander, 21
mailing list, 22
MAPMAKER, 16, 28, 34, 40, 85, 89
mapping function, 35, 86
    Fixed, 36
    Haldane, 35
    Kosambi, 35
marker translation, 90
maximum likelihood, 57
missing data, 51
MODEL, 118
MODELS, 123, 127
MS-Windows, 16, 24, 44
    GNUPLOT, 20
    Windows Explorer, 20

Note, 25–27, 31, 33, 34, 36, 40, 56

OPTIONS, 96, 102, 106, 109, 112, 115, 117, 119, 121, 125, 129, 131

permutation test, 46, 54, 56, 60, 64, 69, 71
PERMUTATION TESTS, 123
phenotype, 39
PREPLOT, 129
Preplot
    automagic, 72
    options, 72
printing, 21, 72, 74
Prune, 42, 60
PRUNE, 112
Prune
    interactive menu, 43

Qstats, 49
QSTATS, 115
Qstats
    options, 51
QTL, 14, 37
QTLCART, 96

Rcross, 38
RCROSS, 109
Rcross
    input, 40, 89
    output, 40
REFERENCES, 100, 104, 107, 111, 113, 116, 118, 120, 124, 127, 130, 132
resource file, 26, 75
RESOURCE FILE, 98
Rmap, 32
RMAP, 102
Rmap
    input, 85
    input format, 33, 34
    options, 33, 35
    output, 36, 87
Rqtl, 37
RQTL, 106
Rqtl
    input, 37, 38, 87
    output, 69, 88

sample
    average deviation, 49
    kurtosis, 49
    mean, 49
    skewness, 49
    standard deviation, 49
    variance, 49
simulation
    cross, 38
    gametes, 39
    genetic linkage map, 33
    genetic model, 37
    missing data, 44, 47
    QTL, 37
    random number seed, 25
    recombination, 39
    trait, 40
SRmapqtl, 47, 58
SRMAPQTL, 119
SRmapqtl
    output, 55
standard deviation, 61
stepwise regression, 47, 55
    backward, 55
    forward, 55
    forward-backward, 55

token, 85

UNIX, 16, 19, 23
    install, 20
    man pages, 21
    shell script, 44, 46
USING THE INDIVIDUAL PROGRAMS, 100

variance
    environmental, 38, 39
    genetic, 39
verbosity, 26
virtual marker, 69

web site, 21, 75
window size, 58
working directory, 24, 26, 75
WORKING DIRECTORY, 99

Zmapqtl, 47, 56
ZMAPQTL, 121
Zmapqtl
    model, 58
    option, 59
    output, 61
    virtual marker, 58